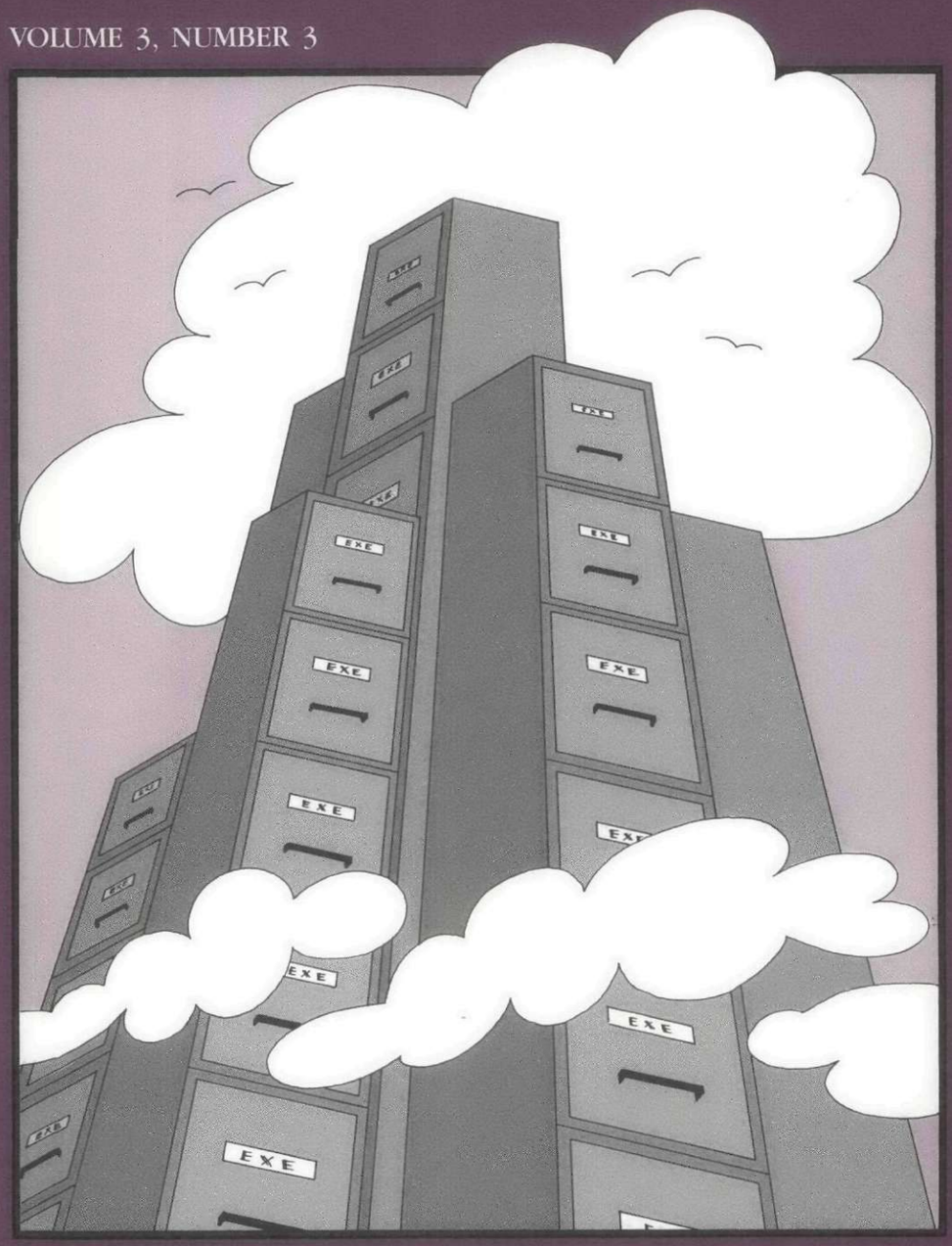
$4.00

programmer's journal

The Resource Journal for IBM PC Programmers

VOLUME 3, NUMBER 3



REDUCING THE SIZE OF YOUR EXE FILES

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Creators of COMPUTER LANGUAGE Sponsor the C Expert Forum

Never before have so many leaders in the C programming field gathered for one event. The C Seminar/Workshop will be an exciting forum on the latest technical innovations and C language developments. Best of all, you'll experience a practical, hands-on approach in small workshop sessions. All this in the beautiful autumn foliage of New England, just four blocks from Harvard Yard. The C Seminar/Workshop is brought to you by the publishers of COMPUTER LANGUAGE.

The cost for this comprehensive 2½ day event is only $695. Sign up by June 30th and receive a $100 early bird discount.

CURRICULUM

Speakers

Jim Brodie, *ANSI C committee chairman:* Overview of the ANSI Standardization Effort
P.J. Plauger, *author, ANSI C committee secretary:* Programming Style and C
Larry Rosler, *ANSI C language chairman:* Language Standardization Issues
Tom Plum, *author:* Efficiency of C Programs
Heinz Lycklama, */usr/group UNIX chairman:* UNIX Perspective on C
Leor Zolman, *compiler writer:* Porting C Programs between Operating Systems
Robert Ward, *C User's Group coordinator:* Structured Methods of Debugging C

Workshops
(Subject to change based on availability)

Seminar participants will be able to choose four from this list:

Debugging Techniques
Interpreters in a Development Environment
Programming for Portability
Efficient Code Generation
Cross Compilers
Network Data Base Theory and C
Object-File Formats for UNIX Systems
Philosophy and Methodology of Benchmarks
ANSI Standards: Questions & Answers
Code Readability and Organization
Asynchronous Communications
Writing Extensions to C
C / UNIX System Subroutine Interfaces
Porting C between CP/M, MS-DOS, and UNIX

C Seminar/Workshop Registration Form

Please enroll me in the C Seminar:

☐ Early Bird $595 (pay by 6/30/85)
☐ Single $695
☐ Multiple
(3 or more enrollments get $100 discount)
☐ I do not wish to enroll at this time but please send me more information.

Method of Payment:

☐ Check Enclosed
☐ Bill My Company

Make check payable to:
C.L. Publications Inc.

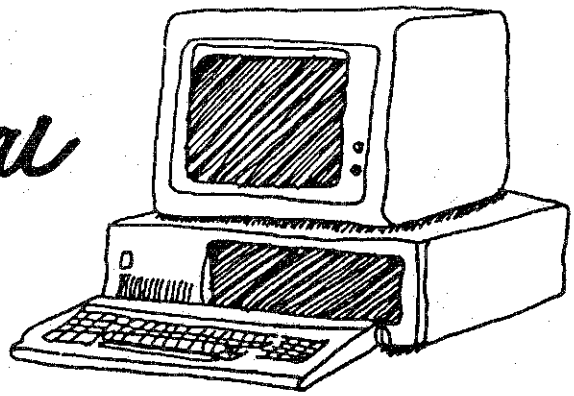
Name & title _____
Name & title _____
Name & title _____
Company _____
Address _____
City, State, Zip _____
Phone _____

85

COMPUTER LANGUAGE Seminar
131 Townsend St.
San Francisco, Calif. 94107
(415) 957-9353

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

programmer's journal



CONTENTS

HOUSEKEEPING 2

PJ FORUM: NAPLPS AND THE PROFESSIONAL PROGRAMMER by Richard Larratt 3
A nuts and bolts look at NAPLPS for software developers

REDIRECTIONS: REPORT FROM UNIFORUM by Frank D. Greco 6
Was it worth all the hoopla?

C: INSTANT-C review by J. Edward Volkstorf, Jr. 13
The new wave of C development is here

ASM: PRINTER CONTROL FROM A BATCH FILE by Michael L. Hoyt 19
Using an assembler routine to control your output

BASIC: REPOSITIONING THE STARS IN STARFINDER by Melvin O. Duke 33
How to approach a three dimensional problem

PASCAL: CONFESSIONS OF A PASCAL PROGRAMMER by Daniel J. Jones 37
Reducing EXE File Size Through Data Positioning

TOOLS: TOOLS FOR BETTER PROGRAMS by Terry W. Cox 40
Setting up your ultimate development environment

BOOKS: SYSTEMS IN ORGANIZATIONS, BUGS + FEATURES review by J. Edward Volkstorf, Jr. . . . 45
New ways to implement systems in different organizations

MISC 47

ISSN# 0747-5861

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

HOUSEKEEPING

The BIG MONEY SOFTWARE IDEA OF THE MONTH goes to one of our fellow readers of PJ. It is a carefully engineered (REVERSE engineered, mind you) product. If you promise you won't steal the concept, I'll describe the way it works. We are half-way through the development cycle already, so we're not too worried. The software idea is absolutely ingenious in a world gone multi-mad.

There are a number of multi-tasking, windowing products like TopView, DESQ, GEM, Concurrent PC-DOS, PolyWindows, APX, etc. Obviously, once you have too much competition, you need to move to the next hot software concept that users are dying to buy. Well, where do we go from here?

You got it. A software package that integrates multi-tasking, windowing environments! Then you will be able to run TopView, DESQ, GEM, and Microsoft Windows all at the same time on your PC. I guess we caught you with your pants down on this one. What more could anyone ask for? You can survive the "I'm better" multi-debates by loading everything in at once. Imagine running GEM in a TopView partition which is crunching away as a subtask under Concurrent PC-DOS being tiled by MS Windows. Your machine will explode with the combined computing power. You could have hundreds of WordStars running, each one editing multiple files. Maybe you could even have every data file on your hard disk open at the same time! When was the last time you did that? WOW.

And here's the punch line. This "secret" software package that integrates every existing multi-tasking product, will appear to the operator as a single user, single task system!!! A truly fabulous concept. First you load the "secret" package, then spend 15 or 20 minutes loading in the multi-devils, at which time the "secret" system takes charge and reverts to the appearance of a single user, one task environment. Pretty neat, huh. Much simpler to use and easier for novice computerphiles to understand.

I know what you are thinking. The "secret" package is just going to spin the disk drives and turn on the red light and appear to load the multi-monsters, without actually doing so. And then once the charade is over, it will revert to kosher PC-DOS.

Now I'm not going to answer yes or no to that question. You see, we are selling the PERCEIVED value of the software to customers. How it does what it does is a programming issue. The customer feels that they are up to date, have the bull by the horns, are on the cutting edge of technology, AND they can understand the final operating environment. Look . . . we are only asking $395 for the package, and the perceived value is enormous. The other companies can't seem to find the pulse of the market. They offer a $100 package that the end user is lost in. What good is that?

No, this was NOT reverse engineered from the marketing department. The tone of your question is verging on the sarcastic. Enough said. We have a new issue of PJ to enjoy!

— Greg □

programmer's journal

Publisher:	Elizabeth Oakley
Editor:	Greg Estes
Managing Editor:	Erin O'Connor
Copy Editor:	Marilyn Reaves
Production:	Lisa A. Davis
Cover Design:	Marilyn Reaves Lisa A. Davis
Illustrations:	Julia King
Contributing Writers:	Richard Larratt Frank D. Greco J. Edward Volkstorf, Jr. Michael L. Hoyt Melvin O. Duke John Jackson Daniel J. Jones Terry W. Cox FNE Warren Hillman

Published by:	Oakley Publishing Co.
Business and Editorial Offices:	Programmer's Journal P.O. Box 30160 1897 Garden Ave. Eugene, OR 97403 (503) 484-2162
Subscription Inquiries:	Programmer's Journal P.O. Box 3000 Dept. EE Denville, NJ 07834 (503) 484-2162
Postmaster send address changes to:	Programmer's Journal P.O. Box 3000 Dept. EE Denville, NJ 07834

IBM and Personal Computer are trademarks of International Business Machines, Armonk, New York; dBASE II is a trademark of Ashton-Tate.
Published bi-monthly.

Copyright © 1985
by Oakley
Publishing Co.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

NAPLPS AND THE PROFESSIONAL PROGRAMMER

by Richard Larratt

MacPaint on the Macintosh and ColorPaint on the IBM PCjr are great showroom drawing programs. Thanks to the excellence of the "novice" interface, a lot of folks who never thought they could do anything useful with a computer discover they can draw pictures with a computer's help.

Beyond the excellence of the novice interface of either program, both share one common problem: file size. In the case of ColorPaint it is always 28,672 bytes/picture, no matter how much or how little was actually drawn (since ColorPaint saves almost the entire 32K 16-color video memory on the junior). MacPaint is better in this department since one "MacPaints" a 99,840 dot window at a time. If you are familiar with MacPaint, you know that a MacPaint screen prints out as a 5 by 3.5 inch window on an 8.5 by 11 inch page.

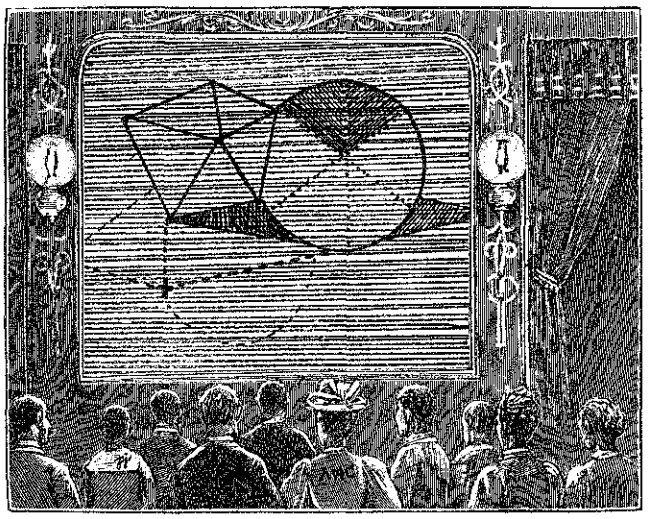
MacDraw is different. It stores a circle as a circle. The geometry of the shape (a circle, for example) and its attributes (color, fill, etcetera) are stored in memory, rather than the individual attributes of the individual dots or pixels as seen on the screen. In programmer's talk, MacDraw is object-oriented rather than bit-mapped. An obvious advantage of MacDraw's over MacPaint's approach is that a video screen can be stored in far fewer bytes of memory and hence far more screens can be stored (manipulated) on a single diskette.

I suppose there is no technical reason why a good programmer cannot write private or "company" graphic subroutines and, thanks to the data compression thus achieved, produce an adventure game or courseware product with more graphics than could be achieved storing video memory. I don't recommend this approach, MacDraw notwithstanding. Why reinvent the wheel?

BRIEF HISTORY OF NAPLPS

A number of years ago the military was faced with the problem of how to communicate graphic video screens, war maps, etcetera. The technique developed in North America and now the basis of AT&T's videotex standard is called NAPLPS, North American Presentation-Level Protocol Syntax. Stated briefly, an image is represented by a file of PDIs, Picture Description Instructions. The major PDIs are point, arc, polygon and rectangle. If you stop and think about it, any computer graphic could be decomposed to a file of these PDIs.

A number of devices that do just that decomposition were demonstrated at Videotex '84 in Chicago; i.e., scan a picture and produce a file of PDIs when processed by a PDI decoder reproduce the picture. A simpler alternative is composing the



picture in PDIs in the first place. Three years ago this would have meant using a special-purpose PDI page creation unit; i.e., as the artist draws the picture, the unit makes the PDI file. Both Norpak of Ottawa, Canada, and AT&T make such $20,000 plus units for professional page creation.

While you won't likely see such page creation units for sale at your local ComputerLand store, most NAPLPS videotex pages have been drawn with either the Norpak IPS-2 or the AT&T unit. I have been consultant to a number of corporations or government departments with as many as 40 such units in a single work area.

EARLY NAPLPS PRODUCTS FOR THE IBM PC

Programmers familiar with the history might remember adverts as far back as 1982, claiming all NAPLPS was possible on an ordinary IBM PC. Quoting from a pioneer advertisement, page 250, *PC Magazine,* January 1983:

"TELIgraph . . . Typical transmission time per screen: 15 seconds at 300 baud . . . 300 – 500 screens can be stored on a 320 Kb diskette . . . Graphic Editor — Decoder — Communications — Slide Show . . ."

But as we all know, it is now 1985, and such products are only now emerging; in the case of TELIgraph, it has changed into HELIOgraph — the promise is the same.

STANDARD PROBLEMS

A problem — no criticism of early entrants such as TELIgraph — is that the videotex standard was in flux until a few months ago, having begun its public life as Telidon 699, then Telidon 709 and now finally . . . T500 NAPLPS.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

This might be a good place to introduce the one reason (a la MacDraw) to not (as a developer) incorporate a standard in your product. The trouble with a "standard" is that no single writer/company/country can control the thing.

Going back in time, imagine a 1982 "699" level of graphic programs doing all the good graphic things promised — like TELIgraph — but within a "closed" universe. The system works just fine coding and decoding PDI files into graphic images but only if the population is "closed", i.e., other similar PCs. When someone buys MacDraw (or WordStar) this is understood. Few customers expect that a WordStar file will exchange with one from VolksWriter.

The history of the "NAPLPS for the (usually IBM) PC" was punctuated by customer expectation that (technically) he/she would be able to access any NAPLPS Videotex trial, be it the TIMES MIRROR test in Rancho Palos Verdes or the Knight-Rider test in Florida.

As one who lived through this history, suffice it to say that it took a lot of shouting and screaming to learn to stick to the same standard. There is now an agreed standard for NAPLPS (I think!).

NEW!

Advanced Trace86™

Symbolic Debugger & Assembler Combo

- Full-screen trace with single stepping; Even backstepping!
- Write & Edit COM & EXE programs
- Conditional breakpoints (programmable)
- Switch between trace and output screen; Or set up two monitors
- 8087, 80186, 80286, 80287 support
- Write labels & comments on code
- Polish hex/decimal calculator
- and more . . . Priced at $175.00

To order or request more information contact:

Morgan Computing Co., Inc.

P.O. Box 112730, Dallas, TX 75011
(214) 739-5895

NAPLPS I HAVE TESTED

As a contributing editor of *Videotex World* (a Bell Canada Publication), I have had the use of a professional page creation system for my IBM PC the past year. As a programming aide, it also runs on my junior with the MicroSoft Booster Card (even though I develop on the PC). The junior is at most a "target" machine for applications. The system I am referring to is the PC Picture Painter from CableShare Inc. PC Painter sells for $3600 (software only). CableShare is in the business of selling complete videotex/videodisc systems; I'd suspect that most PC Painters are sold as part of a larger sale or as an alternative to a single-purpose PDI page creation unit.

CableShare report, that they have sold over 150 PC Picture Painters in the last fiscal year. In their market niche the issue is rarely the user-friendly cutesy stuff that sells MacPaint to the civilians but rather how well and how fast a professional commercial artist or draftsperson can work with PC Painter versus the Norpak IPS-2 or the AT&T units. (The economics of professional usage demand that the PC Painter be as productive as the $20,000 dedicated units.)

While I cannot comment on the AT&T unit (not popular in Canada) I can opine that the PC Painter is the nearest thing to an IPS-2 without being an IPS that I've found! Most PC Painters are used for nothing but page creation. In my own case, page creation is a minor task for my IBM PC; hence the fact it is "software" is an overwhelming advantage over a dedicated unit.

THE DECODER

A complete NAPLPS package consists of page creation, communications, slide or picture management, and the decoder. In my opinion the most difficult part of any NAPLPS package is the decoder. The decoder must be able to interpret a PDI file in a time-dependent mode (perhaps in a communications context at 19,200 baud) and produce a picture. There were some early decoders written in BASIC, but so far as I know they never exceeded 1200 baud.

PC Painter is a solid, year-old C product. It has to be to satisfy its professional users! ProDraw, another drawing package, also in MicroSoft C, is not nearly so old or so solid. On the other hand, ProDraw is $350, not $3600. If you can get Waese to show you the source code, you might be able to live with his bugs for the difference! ProDraw is a "MacDraw" like interface, but not so cutesy. Both use the FBN NAPLPS decoder.

JUST THE FBN DECODER

Depending on what you have in mind, you might want to buy only the FBN decoder and incorporate it into your own product. FBN sells a demonstration diskette for their NAPLPS decoder for $20. Actually, it is not a demonstration diskette. It is the decoder. What you get for $20 is five or six different

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

versions of the decoder for various computers and/or graphics cards. The IBM junior and the IBM PC with various graphics cards are covered (Plantronics, Conographics, etcetera). What you get for another $180 is 16 pages of instructions on how to use their decoder with your code. If you can understand 8088 assembler, you'll have no problem with their instructions. If you beg, they'll give you an "easy" connection from BASIC or PC/Forth. The demo is not copy protected.

"Whenever an application program uses the FBN NAPLPS decoder, three steps should be followed: initialization, invocation and termination." (opening of the 16-page explanation)

The decoder requires 43 Kbytes of contiguous memory, but once incorporated, the detailed 16-color graphics that previously cost 32K per full screen will average out to 2K. Since FBN have positioned themselves as programmers for programmers, they are open to negotiation on run-time fees if you want to consider using their product to get a hundred or more graphic screens on a single diskette product. The two authors are Joe Rosa and Alexi Nicolae. They wrote it in PC/Forth. They've got five man-years into it now.

If you are thinking of any product that is going to need a lot of graphics, you owe it to yourself to at least look at the FBN demo diskette — even if you will need more than a decoder — since it includes a super NAPLPS Slide Show.

In the interests of conserving space I have assumed you've read early articles (1982, 1983) in Byte or elsewhere, introducing NAPLPS. One of the areas where NAPLPS really shines is animation. Whether it's to be HELIOgraph or the FBN demo (being "T500", each can decode any other's PDI files) you must see an animation sequence . . . a snail eating a plant, becoming a butterfly and flying away . . . That particular sequence is a PDI file of just 9600 bytes!

PC Painter
CableShare Inc.
20 Enterprise Drive
London, Ontario, Canada N6A 4L6

ProDraw
Linicon Inc.
144 Hampton Ave.
Toronto, Ontario, Canada M4K 2Z1

HELIOgraph
Microtaure Inc.
P.O. Box 383, Station "A"
Ottawa, Ontario, Canada K1N 8V4

FBN Software
331 Cooper Street, Suite 400
Ottawa, Canada K2P 0G5
(demo decoder diskette: $20.00)
Phone 613-238-1761
Joe Rosa, Alexi Nicolae □

MasterFORTH
Portable programming environment

Whether you program on the **Macintosh**, the **IBM PC**, an **Apple II** series, a **CP/M** system, or the **Commodore 64**, your program will run unchanged on all the rest. If you write for yourself, MasterFORTH will protect your investment. If you write for others, it will expand your marketplace.

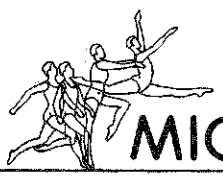
IBM™

MasterFORTH is a state-of-the-art implementation of the Forth computer language. Forth is interactive – you have immediate feedback as you program, every step of the way. Forth is fast, too, and you can use its built-in macro assembler to make it even faster. MasterFORTH's relocatable utilities, transient definitions, and headerless code let you pack a lot more program into your memory. The resident debugger lets you decompile, breakpoint, and trace your way through most programming problems. A string package, file interface, and full screen editor are all standard features.

CP/M™

MasterFORTH exactly matches the Forth-83 Standard dialect described in *Mastering Forth* by Anderson and Tracy (Brady, 1984). The standard package includes the book and over 100 pages of supplementary documentation.

MasterFORTH standard package	
Macintosh	$125
IBM PC and PC Jr. (MS DOS 2.1)	125
Apple II, II+, IIe, IIc (DOS 3.3)	100
CP/M 2.X (in several formats)	100
Commodore 64	100
Extensions	
Floating Point (1984 FVG standard)	$40
Graphics (Apple II series)	40
Module relocator (with utility sources)	60
Printed source listing (each)	35
Publications	
Mastering Forth (additional copies)	$18
Thinking Forth by Leo Brodie	16
Forth-83 International Standard	15
Rochester Bibliography, 2nd ed.	15
1984 Rochester Conference	25
1984 J1 of Forth Appl. & Res. 2(2)	15
1983 FORML Conference	25



MICROMOTION

12077 Wilshire Blvd., #506
Los Angeles, CA 90025
(213) 821-4340

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

REPORT FROM UNIFORUM

by Frank D. Greco

Uniforum, the week-long granddaddy of all the Unix exhibitions and conferences, was held this year in Dallas, Texas. Sponsored by the commercially oriented Unix "/usr/group" from the West Coast, this annual show marked the opening of Dallas' Infomart, the first computer equipment supermarket. While the exhibitions were located on the first two floors of the giant, multi-level computer mall (which was still under construction during the show), conferences were held across the highway at the huge Loew's Anatole Hotel. Also, the academically oriented group "Usenix" coincidentally had their conference a few miles away in downtown Dallas at the five-star Fairmont Hotel.

Frankly, I was disappointed.

At last year's Uniforum in Washington, D.C., Mike Florio, then an executive member of /usr/group, projected that attendance for the 1985 show would be about 35,000. It appeared to me that the week's attendance in Dallas was less than the attendance in D.C. (slightly over 9,000). Is interest in Unix waning? I don't believe so. It is that Dallas was just the WRONG place for the conference. While it is true that in Texas you have the Radio Shack people (more Unix boxes sold than anyone else), and located close by in Irving, Texas, you have IBM's PC/IX and VM/IX group, significant Unix development is done on the EAST Coast or on the WEST Coast. The low attendance at Uniforum does not bode well for Unix vendors. Let's face it, if the general computing public interprets the publicized low attendance as flattening Unix attention, they are not going to be seriously interested in either Unix hardware or software. How many articles on Uniforum will point out my observation that the conference location was a geographical mistake? I say none; yet it is an important issue that affects the Unix market as a whole.

Next year's Uniforum, February 4-7, 1986, will be held at the Anaheim Convention Center in Anaheim, California. Being relatively close to Silicon Valley, this show could be a pivotal one for the Unix industry. If this show is marked by low attendance despite the proximity to Silicon Valley, interest in Unix WILL markedly decrease; while, if many people attend, Unix will get a tremendous boost.

UNIFORUM TUTORIALS

Several valuable tutorials were offered by the sponsors of Uniforum, /usr/group: C language programming, Shell programming, Unix Internals, Interprocess Communication, Lex and Yacc, and other topics. Even though they were delivered concurrently, I was able to attend one of the advanced tutorials presented on the first day. However, for $225, copies of the viewgraphs were NOT made available for the participants. There weren't even large pads available for note-taking! Attendees received only tiny 2" by 4" memo pads. You know the species of pad; they are typically found next to a telephone in a hotel room. In addition, the viewgraphs used in most of the tutorials were black-and-white typewritten (some handwritten!) pages. It is very hard to read typewritten viewgraphs from the back of a room full of 200 people. Free coffee was available only in the morning from 9:00 to 9:30. To top it off, the tutorial was not very good at all. It was not advertised correctly, and it was apparent that the instructor was not well prepared.

Fortunately, I had the opportunity to attend an equivalent tutorial at the Usenix conference down the road at the Fairmont Hotel. The instructor was excellent. Instead of being dry and merely lecturing, he led a very interactive class into the innards of an advanced topic. A very usable folder/clipboard with a regular pad and pen was given to each attendee. Coffee and the usual amenities were constantly available, free of additional charge. This tutorial was well done. The /usr/group people would be well advised to contact the Usenix people for next year's tutorials.

UNIFORUM CONFERENCES

The opening day plenary was given by Dr. Robert L. Carberry, Vice President, Systems Architecture and Technology for the Entry Systems Division of IBM. Dr. Carberry talked about the future of Unix as it relates to IBM. He talked of compatibility between systems and how manufacturers should strive for

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

WHAT THE NEW YORK TIMES SAYS ABOUT

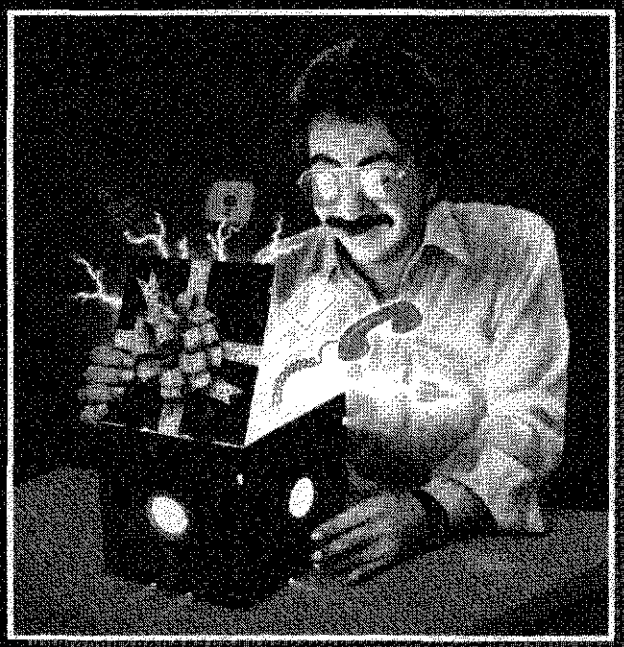
FREE SOFTWARE

"Software does not have to be expensive. In many cases, a program just as useful as the one that costs $495 at the local computer shop may be available for the price of a single floppy disk through public domain libraries.

"But how does one go about finding this treasure of free software, including word processors, spelling checkers, data bases, accounting packages, spreadsheets and above all games, games, games? A smart place to start is with FREE SOFTWARE by Tony Bove, Cheryl Rhodes and Kelly Smith.

"The authors give clear advice on how to get the software and how to use it. They also perform a valuable service by sifting through mounds of junk programs that clog the public domain, saving the user time as well as money."

AN ALTERNATE SELECTION OF THE MACMILLAN SMALL COMPUTER BOOK CLUB



Available now in stores near you!
55948-6 • 320 pp. • $9.95

Distributed by Simon & Schuster Mass Merchandise Sales Co.
1230 Avenue of the Americas • New York, N.Y. 10020

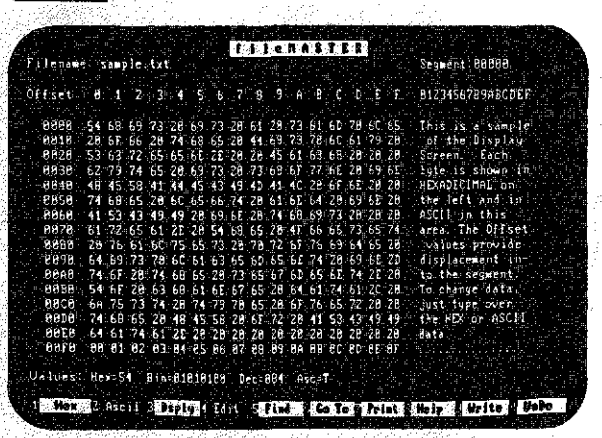
TONY BOVE, CHERYL RHODES, AND KELLY SMITH

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

compatibility. I was ready to call Ripley, believe it or not. What was striking about Dr. Carberry's plenary, which was given to the entire conference, was that he never mentioned System V. He mentioned 4.2 BSD. As a matter of fact, he had several color slides with the Berkeley version of Unix listed. He had one slide with just the text "4.2 BSD." Curious, isn't it? It could mean that IBM will support 4.2 BSD on their machines, or it could be another red herring from the blue meanies trying to add confusion to the whole Unix mess.

I attended many of the Uniforum conferences. They were all BORING! Where was the excitement, the pizzazz? First, I attended "Unix Evolution as It Relates to Office Systems" with John Mashey (one of Mashey's claims to fame is that he designed the Unix shell that predated the now-standard Bourne shell) and Ted Dolotta of Interactive Systems (the PC/IX people). Well, instead of talking about office systems, Mashey and Dolotta talked about the history of Unix from a technical point of view. It was quasi-interesting, but half the audience walked out. It was clear that the audience came to hear about office systems and Unix. Although Mashey (who now works for MIPS Computer Systems, a RISC-based startup) is an excellent speaker, he strayed from the mainstream enough to discourage many people.

fileMASTER THE DISK UTILITY



FINALLY!! **A Disk Utility for the IBM-PC that's fast and easy to use.**

- FILE MODE
- DISPLAY
- PRINT
- MODIFY
- SECTOR MODE
- BROWSE
- SEARCH
- DOS 1.1 to 3.0

ONLY **$39.95** + $2.00 SHIPPING, CA + 6.5% TAX

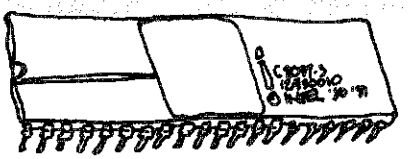
J. L. SCHULLER & ASSOCIATES (818) 366-6934
14800 Rinaldi St., Suite 6, Mission Hills, CA 91345

A side note: Mashey and Dolotta, who both worked on Unix during its infancy at Bell Labs, share the common belief that AT&T unconsciously gave Unix away to universities rather than doing so as a strategic move for the future. Apparently, according to these gentlemen, Bell Labs or AT&T had NO idea that Unix would ever be so popular. It was purely for academic purposes that universities obtained Unix back in the early 1970s.

Another conference session dealt with "4.2 BSD vs. System V for Office Systems." Here Mashey and Dolotta delivered the same talk they gave in the previous session. Dolotta even remarked that all he did was change the title viewgraph. There were two other gentlemen (academic types, really) who talked tedious, monotone, and boring nonsense concerning the superiority of 4.2 over System V. The only bright spot here was Bill Joy's talk. Bill Joy was the main designer of the Berkeley version of Unix. He also designed and implemented the popular ex/vi editor and the C-shell command interpreter. Obviously prejudiced towards 4.2 BSD, Joy stabbed at System V and MS-DOS with some rather subtle and interesting quotes: 'The office does not want the best technology"; "MS-DOS has the same capabilities of Unix version 0 on a PDP"; and "It is not clear that System V will break the MS-DOS hold."

Another session was "Unix and MS-DOS," where Bob Blake (IBM), Adrian King (Microsoft), and Ben Laws (consultant) discussed the two operating systems. This session was one of the best attended. After Mr. Laws spouted some nonsensical drivel on having the printf library call (a common output function used in C) converted to a system call and shared by all users, Adrian King got up to speak. A highlight here was when someone asked Adrian King (Xenix manager for Microsoft) why the directory delimiter under DOS 2.0 and above is by default the backslash. Adrian smiled and paused for a few seconds and said, "Bill will be giving a talk this evening. I think you should really ask him." He was referring to Bill Gates who was also at Uniforum. Rumor has it that IBM did not want PC-DOS to be compatible with Unix.

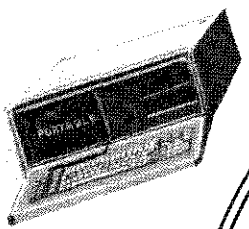
Now Bill Gates was not really listed as a speaker in the Uniforum program, but an informal list of after-hours-birds-of-a-feather meetings (like SIG groups) showed Bill giving a talk at 7:00 p.m. This was at the same time that IBM was giving a talk on Xenix on the PC. There was no comparison in attendance; Gates had a small auditorium to himself while IBM had a small classroom. It was the best attended talk of the



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

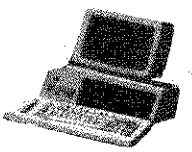
AT LAST!
AN 8086/8088 BOOK THAT TAKES YOU ALL THE WAY!
FEATURING THE IBM PC

INTRODUCING

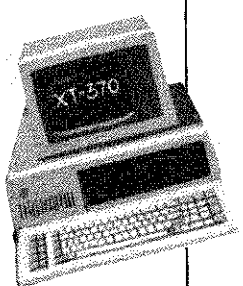


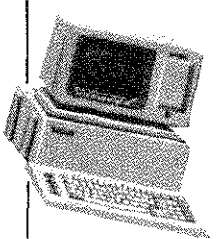
THE SERIOUS ASSEMBLER

A PORTABLE USERS GUIDE



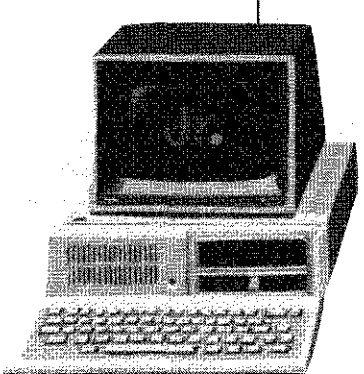
CHARLES A. CRAYNE AND DIAN GIRARD

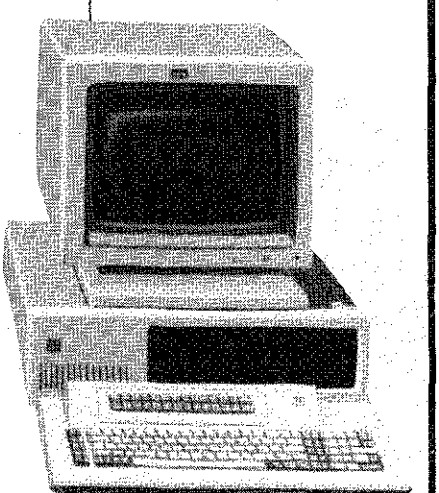




Just how serious are you about
programming in assembler?
If all you're after is a book
that will take you through writing
and assembling Program HELLO.COM,
so that you can say
you've programmed in assembler, maybe this
book offers *more* than you're looking for.
Try that one over there,
the one with the cute cartoons.
But if what you want to do is write
serious programs and subroutines
that use the tightest, fastest possible code
then you've come to the right shop.

If you've got the right stuff, then
THE SERIOUS ASSEMBLER has
the right stuff for you!





MAY 55963-X · 320 pp. · $8.95

BRADY BOOKS

**A selection
of the Macmillan
Small Computer Book Club and
the Library of Computer
and Information Sciences**

Distributed by Simon & Schuster
Mass Merchandise Sales Co.
1230 Avenue of the Americas
New York, N.Y. 10020

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

conference. This is very impressive because the talk was only informally scheduled (a note tacked to the messages board) and it was held after the working day! To top it off, half of the talk concerned MS-DOS and not Unix.

Gates spoke of Xenix's superiority to all other versions of Unix because of its growing compatibility with MS-DOS and PC-DOS. Also, he spoke of the announcement that the Xenix will soon be System V compatible (I will believe it when I see it). And if you had $345 in your pocket, you could attend a Xenix symposium on Thursday through Saturday in Dallas.

Something that was very intriguing in Gates' talk was his mention of the Intel 80386 (iAPX 386). Gates said that, while the 80286 chip is a powerful engine for running PC-DOS and MS-DOS, it has significant compatibility problems with older 8088 programs running in protected mode (as opposed to system or real mode). Most of the audience knew that fact already; PC-Week made sure of that. Gates continued by saying that Microsoft has its eyes set on the 386 chip for running Xenix and MS-DOS simultaneously. According to Gates, the 386 chip does not have the compatibility problems with older PC-type programs. And Gates added that the 386 chip has paging, which should facilitate a standardized implementation of virtual memory.

Also, Gates said that soon MS-DOS will become multi-tasking, and you will see a new group of powerful programs for the PC-type machines. I personally feel that multi-tasking MS-DOS has the potential of pushing out Unix on small machines (see last Redirections column).

In the audience, listening intently to every word, was Berkeley architect and current Sun Microsystems Vice President of Research and Development, Bill Joy.

All in all, the remainder of the conference sessions were very boring except for the highlights which I have mentioned. If Unix is to enjoy commercial success, Uniforum speakers should have some zip to their talks, some professionalism to their viewgraphs (many misspellings), and some neatness to their clothes (many speakers wore no ties and had sneakers on). The conference sessions did not generate much enthusiasm at all.

An added note: /usr/group will no longer be publishing proceedings of the Uniforum conferences as there is a lack of contributing authors. The 1984 conference in Washington, D.C., was the last time that proceedings were published.

EXHIBITOR HIGHLIGHTS

Sritek, the manufacturer of those plug-in processor boards for the IBM PC, surfaced again at Uniforum. Apparently, the business that they have had with IBM kept them extremely busy for the past year. IBM was so interested in their boards that they supplied MIT with PCs equipped with Sritek's 32000 (National Semiconductor) coprocessor boards (IBM also supplied MIT with hi-res graphics boards from an unknown source). Sritek's demo was very impressive. There was an IBM PC/XT with a 32000 coprocessor board running Berkeley 4.1 Unix driving four other PCs (each with terminal emulator packages). All the while, there was PC- DOS (a real PC-DOS!) running in the background. A flip of a key, and Unix was put in the background (still running), and presto! here was PC-DOS graphically solving the Towers of Hanoi problem. File transfer from Unix to DOS and back was also available so that Unix-massaged data could be fed into Lotus, for example.

THE CONNECTOR

Uniform Software Systems' product, the Connector, is a PC-DOS emulator program that runs as a Unix task. It traps all DOS calls (INT 21) and some BIOS calls and emulates them as Unix calls. This was shown last year; this year they have integrated TopView, IBM's windowing package. I saw DOS running in one window and Unix (actually PC/IX) running concurrently in another window on an IBM PC/AT. File transfer between systems was available. Uniform says that even Lotus runs in a window; I was told that the Connector remaps Lotus output so it can run in a window.

GETTING GRAPHIC

Island Graphics Corporation offers "The Micro Illustrator" graphics painting software for several new machines. A brochure says that Island Graphics offers a series of microcomputer paint programs for the new Atari, Commodore/Amiga (!), and the Radio Shack (what new Radio Shack machine?). Their Solar Paint package running on a Sun Microsystems 68010-based Unix system was very impressive although they kept saying that their new "Popup Palettes (tm)" were so user friendly. Ugh! . . . Popup Palettes . . . ?

I WANNA BE LIKE U(nix) WHEN I GROW UP

Lantech Systems' uNETix was available at the show for $49.00 (that's not a typo!). For $49.00, you can have your own baby Unix running on your PC. You only need two double-sided floppies and at least 320K of memory (pretty minimal, I'd say and pretty cheap for a multi-tasking OS for your PC). However, compatibility is only with Version 7 Unix (circa 1978). If you want windowing, add an additional $50.00. A uNETix version of the Lattice C compiler costs a cheap $200 (includes an assembler), and a version of the popular vi screen editor is available for $100 more.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THE SUN SHINES BRIGHTLY

Sun Microsystems was showing off their new Sun-2 Workstations running Berkely 4.2. The Sun-2/170 color Workstation was extremely exciting with its 640 by 480 resolution and 256 simultaneous colors. The founder of the highly touted Berkeley version of Unix, Bill Joy, works as Vice President of Research for Sun. He is a prime designer behind Sun's new Network File System (NFS). NFS was demonstrated at Uniforum by connecting a Sun, a Gould machine, a DEC VAX 11/780, and a Pyramid RISC machine all in a network. NFS makes the network invisible to the user, that is, your files can exist on any of the machines but its physical location remains invisible to you. This is similar to the mounting procedure in Unix, where different file systems are "mounted" in the file hierarchy and the physical location is relatively unimportant to the user. In this way, a user can avoid addressing a file by preceding the filename with the logical disk name a la PC-DOS.

NEW C JOURNAL

Que Corporation is starting a magazine for C language programmers called "−C". It is available monthly for $60.00, which is extremely steep. Although the sample looked informative, the price is too high for wide circulation.

HEWLETT PACKARD'S TRANSPORTABLE UNIX

HP had a packed booth every day with people interested in the Unix-kernel-in-ROM based transportable. It seemed like an interesting machine, but it still is "floppy based." Unix is a disk hog and needs a fast winchester to be happy. HP could not fit all the Unix utilities on the hard floppies, so many programs are missing. The windowing is very impressive and the screen is reasonably readable. I like the idea of the integral printer; I always liked that idea ever since I used the HP 2621P terminals that had a thermal printer built-in so that you could get a rough copy of your work. I also like the idea that HPUX, HP's version of Unix, has the capabilities to perform some real-time duties. Most Unixes cannot be used reliably for real-time programming, like data acquisition and monitoring, because of the inherent scheduling of processes in Unix. But HP has altered the kernel sufficiently to make Unix recognize real-time interrupts instantly and not merely schedule interrupts asynchronously.

THE PROGRAMMER'S SHOP™

helps compare, evaluate, find products. Straight answers for serious programmers.

SERVICES
- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature free
- BULLETIN BOARD - 7 PM to 7 AM 617-826-4086
- Dealer's Inquire
- Newsletter
- Rush Order
- Over 700 products

Free Literature - Compare Products

Evaluate products Compare competitors. Learn about new alternatives. One *free call* brings information on just about any programming need. Ask for any *"Packet"* or "Addon Packet": ☐ ADA, Modula ☐ "AI" ☐ BASIC ☐ "C" ☐ COBOL ☐ Editors ☐ FORTH ☐ FORTRAN ☐ PASCAL ☐ UNIX/PC or ☐ Debuggers, Linkers, etc.

RECENT DISCOVERIES

FASTER C - Lattice users eliminate Link step. Normal 27 seconds, Faster C in 13 secs. MSDOS $95

"C" LANGUAGE

		OUR PRICE
MSDOS: C86-8087, reliable		call
Instant C - Inter., fast, full		495
Lattice 2.1 - improved		call
Microsoft C 2.x		279
Williams, debugger, fast		call
C Systems & debugger		175
CPM80: EcoPlus C - faster, SLR		275
BDS C - solid value		125
MACINTOSH: Softworks		365
Megamax-object, full		275
Consulair's MAC C		275

Compare, evaluate, consider other Cs

BASIC

	RUNS ON	
Active Trace-debug	86/80	75
BASCOM-86 - MicroSoft	8086	279
BASIC Dev't System	PCDOS	115
BetterBASIC - 640K	PCDOS	185
C8-86 - DRI	CPM86	419
Prof. BASIC Compiler	PCDOS	89
Databurst - screens	MSDOS	215
SCREEN SCULPTOR	PCDOS	115

Ask about ISAM, other addons for BASIC

SERVICE

ALL PRODUCTS - We carry 700 products for MSDOS, CP/M 86, CP/M 80, Macintosh and key products for other micros.

EDITORS Programming

	RUNS ON	OUR PRICE
BRIEF - Intuitive, flexible	PCDOS	195
C Screen with source	86/80	75
Epsilon - like EMACS	PCDOS	195
FINAL WORD-for manuals	86/80	215
MINCE-like EMACS	PC/80	149
PMATE-powerful	8086	185
VEDIT-full, liked	86/80	119

UNIX PC

COHERENT - for "C" users	PClike	475
COHERENT-NCI-Realtime	PClike	call
XENIX - plus C to MSDOS	PC	1275

Ask about run-times, applications, DOS compatibility, other alternatives. UNIX is a trademark of Bell Labs

LANGUAGE LIBRARIES

GRAPHICS: GraphiC-source in C	MSDOS	219
GRAPHMATIC-3D, FTN, PAS	PCDOS	125
HALO-fast, full-all lang.	PCDOS	139
FILE MGNT: BTrieve-all lang.	MSDOS	215
CIndex + -source, no royal.	86/80	369
CTree-source, no royal.	ALL	369
dBC ISAM by Lattice	8086	229
dB VISTA-"Network" Structure	MSDOS	465
PHACT-up under UNIX, addons	MSDOS	225
OTHER: CUtil by Essential	MSDOS	129
Greenleaf - 200 +	MSDOS	159
CSharp - **Real-Time**	MSDOS	600
PORTABLE C to PC, Mac, II	Many	125
SOFT Horizons - Blocks I	PCDOS	139
SCREEN: CURSES by Lattice	PCDOS	125
CView - input, validate	PCDOS	195
MetaWINDOW - icons, clip	PCDOS	139
PANEL - many lapg, term	MSDOS	249
ProScreen - windows, source	PCDOS	415
Windows for C	MSDOS	175

FORTRAN

	RUNS ON	OUR PRICE
MS FORTRAN-86 - impr.	MSDOS	$ 239
DR Fortran-86 - full '77	8086	249
PolyFORTRAN-XREF, Xtract	PCDOS	165

OTHER PRODUCTS

Assembler & Tools - DRI	8086	159
Atron Debugger for Lattice	PCDOS	395
cEnglish - dBase to C	MSDOS	750
C Helper: DIFF, xref, more	86/80	135
CODESMITH-86 - debug	PCDOS	139
MacASM-full, fast, tools	MAC	115
MBP Cobol-86 - fast	8086	680
Modula 2 for	MAC, PCDOS	90
Micro: SubMATH-FORTRAN full	86/80	250
Microsoft MASM-86	MSDOS	125
MSD Debugger	PCDOS	119
Multilink - Multitasking	PCDOS	265
PC/FORTH + -well liked	MSDOS	219
PFIX-86 Debugger	MSDOS	169
PL/1-86	8086	495
Polylibrarian - thorough	MSDOS	95
PolyMAKE	PCDOS	95
PROFILER by DWB - flexible	MSDOS	109
Prolog-86-Learn, Experiment	MSDOS	125
SLK/F - Copy Protection	PCDOS	145
SYMD debugger-symbols	PCDOS	119
TRACE86 debugger ASM	MSDOS	115

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and POs. All formats available.

Call for a catalog, literature, and *solid value*

800-421-8006

THE PROGRAMMER'S SHOP™

128-I Rockland Street, Hanover, MA 02339

Visa Mass: 800-442-8070 or 617-826-7531 MasterCard 8517

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

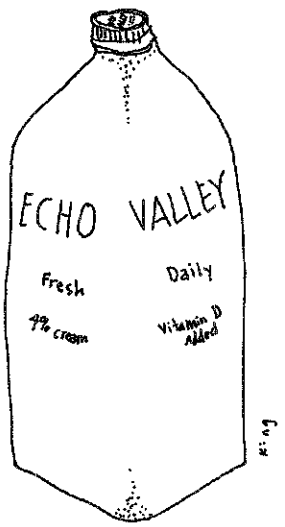
AN IMPRESSIVE MACHINE

Not too many new machines surfaced at the Uniforum show. There was a modicum of new announcements, unlike last year's show. Probably the most impressive machine in the entire exhibit was the Lamborghini Countach (did I spell that right?). Some vendor had a beautiful sleek black Lamborghini in front of his booth, which stopped many passersby. I overheard people talk about it during lunch and dinner. It isn't too often that one sees a car that costs about as much as a VAX 11/750.

DOS BUSTERS?

The 1985 Uniforum show was the first time that many people finally saw the light concerning Unix. Unix will not replace MS- DOS or PC-DOS in the office; I have been saying that over and over again. Unix will have to coexist with MS-DOS. When will everyone else recognize that fact?

The Unix industry needs a better conference than the lackluster one that I attended in Dallas in 1985. □



Cereal Interface

THE HAMMER

Software Tools in C

Designed for the programmer with the user in mind.

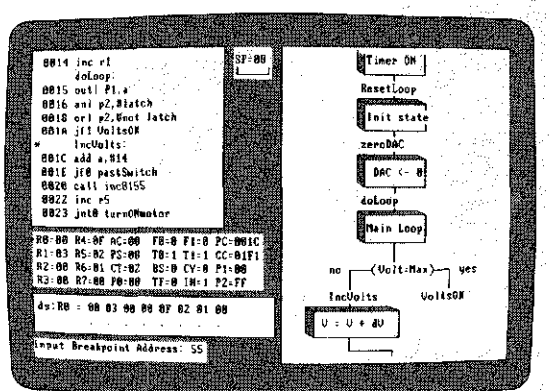
More than just BIOS/DOS access. , THE HAMMER ALSO provides screen attribute control, Smartmodem control, UNIX-like pattern matching, 123-like command menus, date & time processing routines, easy data entry & verification of dates, numbers, and strings, AND MORE. Super data entry routines give the programmer an easy to use, solid interface with the user.
Requires: IBM-PC/XT, DOS2, CI-C86 Compiler
Other C compilers supported soon.

Routines tried & true. $110 with source and manual.
For ordering or inquiring, call or write now:

O E S SYSTEMS
1906 Brushcliff Road
Pittsburgh, PA 15221
412/243-7365

Looking for the right tool for the job?
GET THE HAMMER

Simulator/Debugger for IBM-PC for 8048 - 8051 - 7000 µCs



Execute and debug code for popular single chip microcomputers on your IBM PC. Dynamic display with windows for source code, control flow, registers, flags, memory, commands, and more! Set breakpoints, traps, etc. Cross-assembler & EPROM programmers, too.

	8048	8051	7000	Z8	
X-Assembler	✓	✓	✓	✓	$195
EPROM Prog	✓	✓	✓	✓	$245
Simulator	✓	✓	✓		$395

Demo disk and manual for simulator only $39.50

Cybernetic Micro Systems
Box 3000 • San Gregorio, California 94074 U.S.A.
(415) 726-3000 • Telex 171-135 Attn: Cybernetics

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Instant C "a review"

INSTANT-C

review by J. Edward Volkstorf, Jr.

While testing your C program, an error occurs. An obviously incorrect value is reported on the screen. Is it a problem in that new function you added, or is it your mainline program calling it? Maybe worse yet, it's from some side effect generated in some distant part of your program.

It looks like it's time for a serious session with DEBUG. Or maybe you add a few "printf" functions in strategic places throughout the program. The latter, of course, means some wasted time editing, re-compiling and re-linking the program. If the problem is isolated to a single incorrectly coded function, you might want to re-code it in a separate program to speed up the testing process. In any event, it easily becomes a tedious process.

Now let us reconsider the above scenario with Instant-C, an interpretive C offered by Rational Systems of Natick, Massachusetts. Once the programming error described above shows itself, you call the suspected function directly. Try a few different values and observe the values returned. For example, look at Figure 1, where a dialogue with Instant-C is used in testing a function called "gcd" which calculates the greatest common denominator of two integers. The # is Instant-C's prompter.

The program file named GCD.C is brought into the Instant-C memory space with the load command. As it is loaded, each function is compiled and identified (just main and gcd in this case). Any #defines and #includes would also be listed at this point if I had used them. After loading I can begin testing any function in my program.

I simply type the function name with literal arguments and the result is displayed. Instant-C will bark at me if I call the function with too few or too many arguments or if the arguments are the wrong type. I can even define variables to use, and if I use some not already defined variables Instant-C will tell me. Notice how I can coerce the floating point value 42.33 to integer and also use expressions as in gcd(x+6,30).

Once I have discerned the problem with my function I can quickly edit it, re-test it, and then if satisfied re-test the whole program. What about compiling and linking you might ask? Not to worry — Instant-C takes care of all of this automatically as you exit from editing the function.

Instant-C is a total programming environment for the C programmer. Like Lotus 1 2 3 for the business professional, Instant-C provides a completely integrated set of programming tools designed for the professional C programmer. Instant-C allows you to move easily to and from your C source, the compiled code, library code, run-time results, and intermediate tests — all without leaving the Instant-C environment. It also allows testing individual C language statements and any function from the system or your own function library. There is also an automatic "pretty print" of your source code and a "lint"- like system checker that insures overall system integrity.

Using Instant-C could not be much simpler. You type IC at the DOS prompt, and after Instant-C is loaded its prompt of # is displayed. At this point there are approximately 60 commands available for you to enter.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
# load GCD
main defined.
gcd defined.
GCD.C loaded.
# gcd(125,75)
        25
# gcd(39,52)
        13
# gcd(7)
**ERROR: Too few args in call to gcd.
# gcd(1,2,3)
**ERROR: Too many args in call to gcd.
# gcd(42.8,7)
**ERROR: Incorrect argument type(s) in call to gcd.
# gcd(x,51)
**ERROR: I'm sorry, but I don't know the word "x".
# int x
# x=34 ; gcd(x,51)
        34
        17
# gcd( (int) 42.33, 49)
        7
# gcd( x+6,30)
        10
#
```

Figure 1. Instant-C dialogue example using "gcd" function

```
#
# int x
# x=2
        2
# --x
        1
# x = 3 * ( 4 + 3 )
        21
# x % 6
        3
# toupper('h')
        'H'
# isalpha('4')
        0
# printf("\n\n %s %d %c \n\n","NUMBER IS",96,33);

 NUMBER IS 96 !

# for (x=1; x<9; x++) printf("%d  ",x); printf("\n\n %s \n\n","DONE!");
1  2  3  4  5  6  7  8

 DONE!

#
```

Figure 2. C statement processing examples.

```
Instant-ED Copyright(C) 1984 by Rational Systems, Inc.
mode INSERT    (new function) gcd              line    1 of    3, col    1
int gcd( )
    {                   /* gcd */
    }                   /* gcd */


Instant-ED Copyright(C) 1984 by Rational Systems, Inc.
mode INSERT    (new function) gcd              line    7 of    7, col    2
int gcd(m,n) /* greatest common denominator of m and n */
int m,n;
{
int r;
while ((r=m % n) != 0) { m=n; n=r; }
return n;
}


Instant-ED Copyright(C) 1984 by Rational Systems, Inc.
mode INSERT    function gcd                    line    1 of   11, col    1
int gcd(m, n)           /* greatest common denominator of m and n */
int m, n;
    {
    int r;
    while ((r = m % n) != 0)
        {
        m = n;
        n = r;
        }
    return n;
    }
```

Figure 3. Instant-C edit session examples with gcd function.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
#
#
# list main
void main()
    {
    int a, b;
    puts("Enter two numbers...\n");
    while (scanf("%d %d", &a, &b) == 2)
        printf("\nTheir GCD is %d \n\n", gcd(a, b));
    }
# run
Enter two numbers...

75  125

Their GCD is 25

77  28

Their GCD is 7

#
#
```

```
# trace gcd
# run
Enter two numbers...

39 52

** Execution Interrupted:  entering traced function
    at "gcd" line 3:
            {
# m; n;
        39
        52
# go

** Execution Interrupted:  exiting traced function
    at "gcd" line 11:
            }
returning with value = 13 decimal
# r
        0
# go

Their GCD is 13
```

Figure 4. Program execution and use of trace and go commands.

```
#
# trace gcd
# run
Enter two numbers...

125  75

** Execution Interrupted:  entering traced function
    at "gcd" line 3:
            {
# back
Level 2 condition:  entering traced function
  occurred in function "gcd" line 3:
            {
            int r;
  called from function "main" line 6:
                printf("\nTheir GCD is %d \n", gcd(a, b));
            }
  called from function "_main" line 20:
            main(argc, argv);
            exit(0);
  called from Instant-C command line:
        _main(_command_line);
Level 1 condition:  call to "_exit"
#

# go

** Execution Interrupted:  exiting traced function
    at "gcd" line 11:
            }
returning with value = 25 decimal
# go

Their GCD is 25
#
#
#
```

Figure 5. Back command illustration.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

As an example of individual C code statement execution, look at Figure 2. The variable x is defined, assigned the value 2, and then some expressions using x are entered. Instant-C processes any non-declarative statement immediately, printing the results indented on the next line. Library functions can also be called, and even multiple statements such as the for loop and printf example at the end of the figure.

If you are beginning the development of a new program or function, you simply type "ed gcd" and an edit session begins for the named function (gcd in this case.) For programs coded earlier you first bring the source code into the Instant-C environment with the command "load gcd" as illustrated earlier. You then edit a function within that file. You edit individual functions — not the entire source file — and thus name the function being edited and not the filename.

Figure 3 illustrates a simple edit session for the function gcd. The top third of the figure illustrates how the Instant-C editor creates a framework for you to begin entering your code: the function name is supplied with opening and closing braces, plus two comments.

The function, as I wrote it, is listed in the middle third of the figure. Note how I put the calculations for m and n on the same line after the while statement. The lower third of the figure illustrates Instant-C's "pretty print" of the same function when I later re-call it for further editing: automatic indentation, spacing before and after operators, and a single statement per line.

When finished editing, I exit back to the Instant-C command interpreter, where I can then call and test the function as illustrated in Figure 1. Source code is stored on disk by typing the command save followed by the filename.

You may test your functions or the entire program at any time. Simply type run, and execution begins as though the program had been called directly from DOS. To control execution, breakpoints may be set using the trace command. You simply type trace, followed by the function name. When this function is called, execution will stop and return you to the Instant-C command mode where you can examine variables, review and even change source code. Figure 4 illustrates this further.

Note that in Figure 4 and the next one, commands I enter have been underlined so as to distinguish them from Instant-C responses.

In the example, I list the "main" which calls my function "gcd". I then run the program calculating the gcd of two pairs of numbers. After exiting from the program I set a breakpoint on the gcd function with the trace command and re-run the program. When processing stops in gcd, I can examine the variables passed to it, m and n in this case. Entering the go command re-starts processing. When a traced function is finished processing, another breakpoint occurs and the value of the function is reported. I can then examine any of the variables local to the function, such as r in this case. I enter go to continue processing which returns to main.

If you are interested in the path of execution that took your program to calling this function, you enter the command back. This essentially lists nesting levels and points at which function calls were made that got you where you are. See Figure 5 for an example of the back command.

Should another function be of interest, at this point you may examine it by first entering the command local followed by the function name. This allows you to examine the named function's variables.

I found myself writing some simple C programs in a matter of minutes after learning the Instant-C editor and command processor. A standard test for C is prime-number generation using Eratosthenes' Sieve. I coded a version of this used in *Byte* and other magazines. Instant-C compiled the code in a matter of seconds. Running it for 10 iterations of searching for primes from 0 to 8192 took approximately 35 seconds. This is not particulary impressive since I have read of a number of other C compilers producing code that runs in about 12 to 15 seconds.

To create an .EXE file you simply enter "make" followed by the name of the file. For the Sieve test the resulting file was also quite large at 32K. On the other hand, much of this is due to a large standard library of functions that are included. A later version of Instant-C will supposedly allow selectively including library functions in the compiled code.

The above brings to mind another interesting feature available with Instant-C: the ability to customize the system. This essentially means that you can modify the standard function library that Instant-C itself uses for which the source is provided. A baseline version of IC is provided, called ICBASE. You may use Instant-C to load the modified library, save it with ICBASE, and your very own Instant-C is created. You could add or delete functions or change some of the standard ones to perform differently.

The source code for an extensive C function library is also included, as well as a set of standard IO functions, plus string manipulation, character and type conversion, and memory management functions. Some low-level routines are provided for DOS function calls, port IO, direct memory calls, and memory data moves.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Library source is also maintained while you are interactively using Instant-C. For example, I could examine and edit any of the standard library functions. This should be useful as a kind of "pseudo-help." If for example you are unclear about the calling conventions for a particular function you can simply say "ed malloc," for instance, and see exactly what the function does. The library could even be modified on the fly. I did this with the printf function, changing the hexadecimal format code %x to %h.

Some of the other commands available to you from Instant-C include, most notably:

dir	=	DOS-like directory command
type	=	DOS-like type a file command
erase	=	DOS-like erase file command
llist	=	list your program and function to the printer
listname	=	list the data variables, #defines, #included files and functions in the current file
list	=	list the source code for the named function, data variable, or #defined literal
step	=	resume execution for one statement; this command also has forms that include an in/out/return suffix to allow more than one statement and function to be executed
untrace	=	clear function call breakpoint
delete	=	erase the named function or data variable
new	=	clear the current memory area for a new C programming session
quit	=	terminate your Instant-C session.

There is also a series of commands for displaying memory in a variety of formats: character, decimal, octal, hex or string.

The Instant-C documentation contends that it accepts all the more popular C dialects, although none are explicitly mentioned other than CI 86 when the type unsigned char is discussed. The overall Instant-C language is not defined — in my manual this is Appendix D. A "Language Summary" is indicated as being provided in a later release. Overall the manual is adequate. The lone example is a hello program (printf("Hello, World!—n"). This does a disservice to the system when you consider all the various commands available that a more involved example could illustrate as I have done in this review.

The lack of a full language definition might seem odd, but a little background is now in order. Instant-C as I write this has just undergone its first "public release." The version I tested was 1.01. Earlier versions were sold as pre-release systems, but users were alerted to the fact that "serious bugs" may still be lurking in the system.

Instant-C™: The Best Value In C Programming Tools

The edit-compile-link-test-debug cycle that takes tens of minutes with compilers and linkers is only seconds with the *Instant-C* interpreter. Yet it runs your programs **50 to 500 times faster** than conventional C interpreters! You get the best of both compilers **and** interpreters. Only *Instant-C* is a complete, integrated environment for creating, testing, and running your programs.

Instant-C gives you **all** of these proven capabilities in one tightly integrated package:

interpreter—*Instant-C* runs your programs faster than some compilers; has direct execution; full K&R

compiler—*Instant-C* can make stand-alone programs

full-screen language editor—shows syntax errors with cursor set to trouble spot

C source debugger—single-step, breakpoints, stack trace, more

run-time checker—validates pointer refs, array bounds, more

C source formatter—save editing time, find logical flaws

standard library with source—for best portability

linker—work with multiple source modules

Lint—extensive compile-time validation

The cheapest available examples of these tools would cost $800 (and they don't even work together). You could spend close to $3000 to get the best product of each kind, but you'd have ten times the complexity, filling megabytes of disk. *Instant-C* is faster: it performs these functions automatically. *Instant-C* is far more than the sum of its parts.

Instant-C is all of these capabilities in one package, fits on a single floppy disk, is full K&R, works on IBM PC's, compatibles, and others under DOS or CP/M-86. It costs only $495.

Instant-C is the best value in C programming tools. Guaranteed, or your money back for any reason in first 31 days.

Rational Systems, Inc.

(617) 653-6194
P.O. Box 480
Natick, MA 01760

Instant-C is a trademark of Rational Systems, Inc.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

There was nothing underhanded about all this — Rational Systems admitted it up front to all users and has been using these early sales no doubt to help fund refinement and further development of Instant-C. These pioneering users in turn get in on the ground floor of the development process and have the chance to influence the system's design and its features.

Instant-C is an ambitious undertaking. What comes to mind as I use it is the PC's interpretive BASIC. With BASIC I can code my routines and call them directly with GOSUBs. I set breakpoints with STOP and examine variables by simply PRINTing them. All these features are available with Instant-C, plus the pretty print, compiler, and linker. There's even a correspondence to some commands: LIST, SAVE, and RUN for instance are essentially the same in the two systems.

What bothers me about Instant-C is its heftiness — it requires 320K of RAM and costs $500. Of course the BASIC interpreter is included with DOS for $60, a compiler is around $300, and a good screen editor about $100, for a total of $460. The rub is that all of these are completely separate products and not at all integrated as is Instant-C.

Symbolic debuggers exist for C that allow setting breakpoints and examining variables. For instance, DeSmet C offers an editor, debugger; plus there is a Tiny C from DataLight that has similar features. Both of these have far less overhead than Instant-C, although they are not full Kernighan and Ritchie implementations.

This comes back to the lack of an integrated environment that ties everything together. I guess I could use my RAM disk or set up a few memory partitions using one of the multiprogram systems available. This would then simulate the Instant-C environment, but with all this trouble, why bother when Instant-C has already done it for me? Maybe it's not such a bad deal after all. When you get right down to it, I don't think there's a better way that you could learn C.

Instant-C Rational Systems, Inc.
Post Office Box 480
Natick, Massachusetts 01760
(617) 653-6194 □

(For programmers only...)

Stan's Editor

A full-screen editor for the PC.

Stan's Editor is small, fast, and powerful. It uses all available memory on the PC. (With a 512K PC, you can edit over 430K in memory!) The editor has a split-screen, so you can edit two files, each in a 12-line window, or edit a single file on the full screen. The editor allows very long (up to 65,000 bytes!) lines, with horizontal scrolling.

Stan's Editor has auto-indent (great for C and Pascal), and has a "match this brace (or parenthesis)" command. For assembler, the editor's condensed display shows only labels starting in column one, which provides a summary view of your code. DOS directory pathnames are supported and you can exit to DOS temporarily.

Stan's Editor is written in assembler. It's small (26K .COM file) and *very* fast! Three Help screens are built-in, giving you an online reference card. The editor also has optional word wrap. It is very easy to configure (you can even select your cursor).

Stan's Editor is only $65. Or for just $5.00 (applies toward purchase) our demo disk does everything except save a file.

Demo disk only $5

THE CODE WORKS
5266 Hollister, Suite 224
Santa Barbara, CA 93111
(805) 683-1585

Attention OEMs: we are dealing! Give us a call.

First official portrait of famous AT guru, Baba RAM DOS 3.0, seen here with two of his many devoutees.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PRINTER CONTROL FROM A BATCH FILE

by Michael L. Hoyt

There are several programs on the market that use a menu-driven approach to setting the various parameters of the user's printer. But like many programmers, I use batch files whenever possible and generally don't care for applications that insist on making me spend time responding to a menu query.

Therefore, I decided to write a program that would let me set my printer's parameters from a batch file. I could then just add a line to some of my existing batch files that set-up the work environment for each of several different programming applications and have the printer pre-set for program listings. SET-PRN.COM is that program.

Rather than spend a long time explaining every detail of the program listing (which I hope has enough comments for an acceptable level of understanding), I will just hit on a few points.

MAIN

The MAIN procedure's basic function is to step through the command line, looking for SWITCHes and then passing control to the appropriate code module that handles that particular switch request. The display of error (and exit) messages is also handled as part of this procedure.

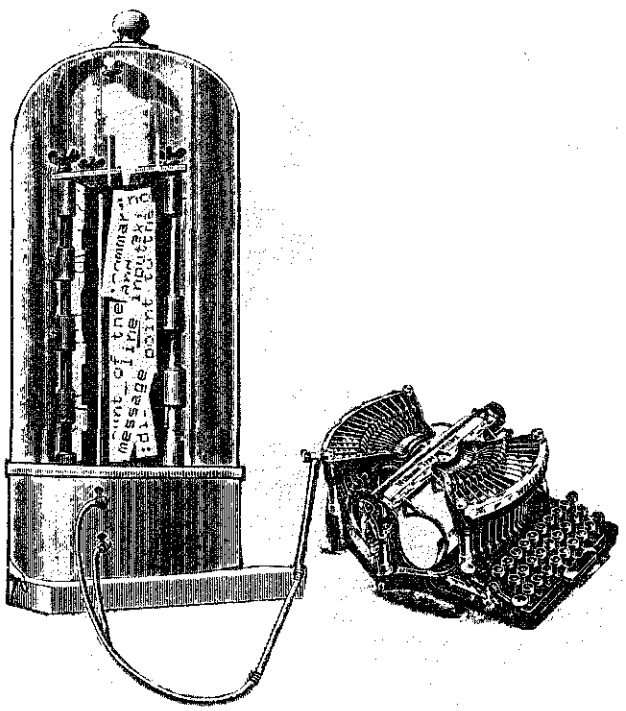
Throughout the program [DI] is used only as a pointer to the user's input on the command line and nothing else. By not using [DI] for more than one function and by taking the effort to keep it up to date, you will find the program logic (such as it is) much easier to follow and will always have the value ready for quick code execution.

Also note that the Carry Flag is used to indicate the presence of an error condition (0 = no error, 1 = error). By keeping [DX] or [BP] pointed at the appropriate error message, checking the Carry Flag is all that is needed before continuing with program execution or displaying a message and exiting, should an error occur.

For the purpose of discussion, we will assume that the user's input is as follows.

SET-PRN /Ed0

The program begins by making sure that the user has indeed included at least one switch on the command line. If input is found (an immediate exit is called for if there is no input), then the next byte after the switch indicator, "/", should be the SWITCH. This byte is then compared to those in Table_1

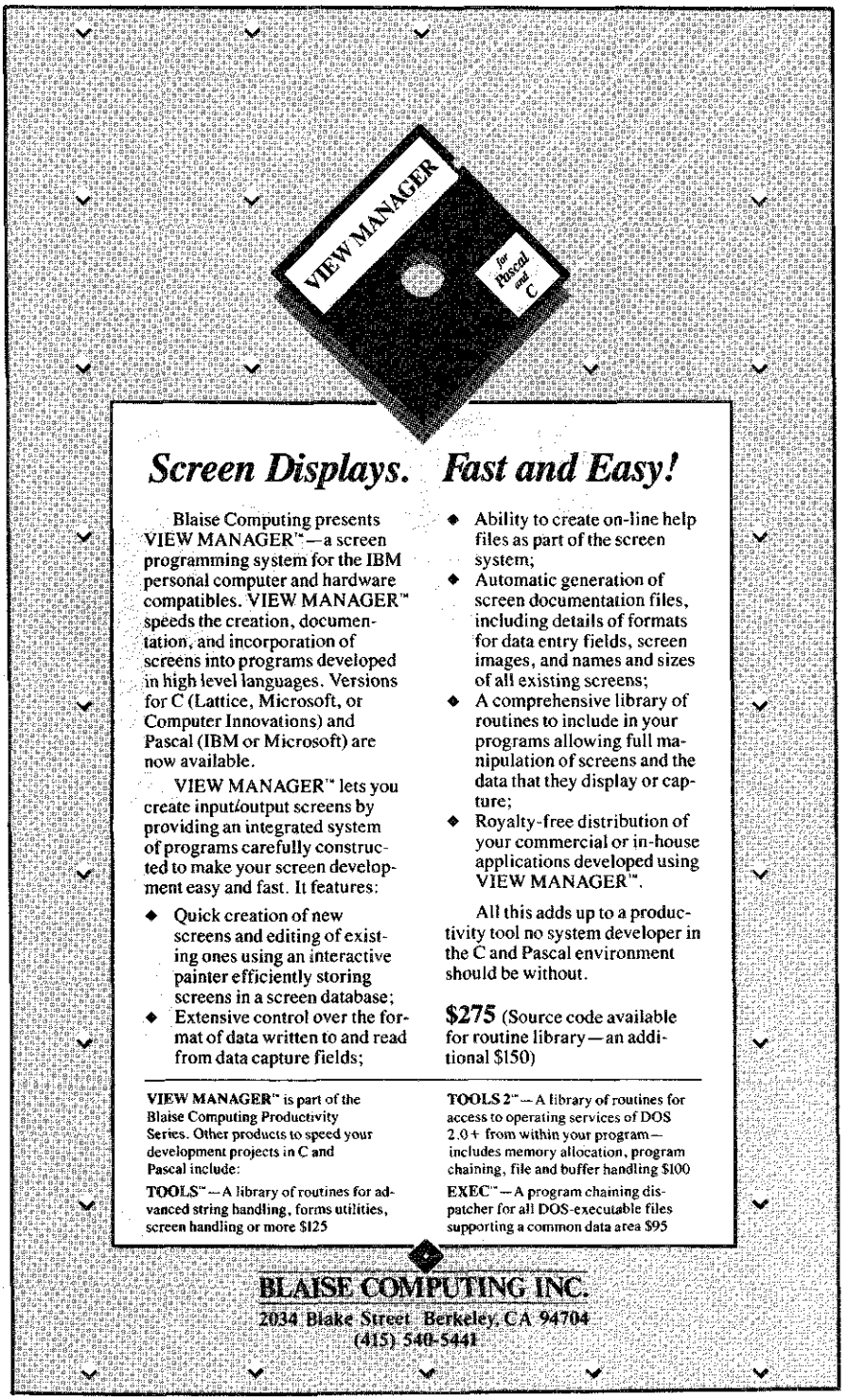


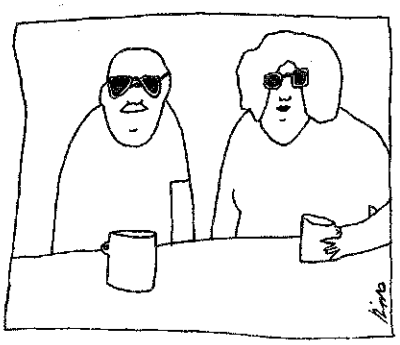
to make sure that it is valid. If the SWITCH is valid, the ASCII value itself is used to direct program execution to the correct module of code to handle the setting of this switch.

The method I have used here for finding the correct module is a very useful technique for directing program execution to subroutines. The idea is to design a look-up table (Table_2 in this case) that contains the addresses of each of our subroutines. The trick is to design the table so that the program has to do minimal manipulation of its current information in order to find the correct offset into the table where the needed address is stored. Table_2 was designed with each capital letter of the alphabet assigned a WORD of storage. That makes it a simple matter of one subtraction and one multiplication (by 2) to find the correct table offset. (An added benefit of this design is that it leaves plenty of room for new SWITCHes by assigning them to the unused letters.)

```
SWITCH:   sub al,41
          shl al,01
          mov bx,offset Table_2
          xor ah,ah
          add bx,ax
          inc di
          call word ptr [bx]
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED


VIEW MANAGER
for Pascal and C
Screen Displays. Fast and Easy!
Blaise Computing presents VIEW MANAGER™—a screen programming system for the IBM personal computer and hardware compatibles. VIEW MANAGER™ speeds the creation, documentation, and incorporation of screens into programs developed in high level languages. Versions for C (Lattice, Microsoft, or Computer Innovations) and Pascal (IBM or Microsoft) are now available.
VIEW MANAGER™ lets you create input/output screens by providing an integrated system of programs carefully constructed to make your screen development easy and fast. It features:
◆ Quick creation of new screens and editing of existing ones using an interactive painter efficiently storing screens in a screen database;
◆ Extensive control over the format of data written to and read from data capture fields;
◆ Ability to create on-line help files as part of the screen system;
◆ Automatic generation of screen documentation files, including details of formats for data entry fields, screen images, and names and sizes of all existing screens;
◆ A comprehensive library of routines to include in your programs allowing full manipulation of screens and the data that they display or capture;
◆ Royalty-free distribution of your commercial or in-house applications developed using VIEW MANAGER™.
All this adds up to a productivity tool no system developer in the C and Pascal environment should be without.
$275 (Source code available for routine library—an additional $150)
VIEW MANAGER™ is part of the Blaise Computing Productivity Series. Other products to speed your development projects in C and Pascal include:
TOOLS™—A library of routines for advanced string handling, forms utilities, screen handling or more $125
TOOLS 2™—A library of routines for access to operating services of DOS 2.0+ from within your program—includes memory allocation, program chaining, file and buffer handling $100
EXEC™—A program chaining dispatcher for all DOS-executable files supporting a common data area $95
BLAISE COMPUTING INC.
2034 Blake Street Berkeley, CA 94704
(415) 540-5441



software pirates incognito

Once the address of the desired module is found, the program calls that particular module to handle the setting of the desired parameter. To illustrate the typical program flow, we will follow the setting of two somewhat different printer parameters. Our first example is to set the printer to double-strike printing, one of the printer Enhancements that can be handled by SET-PRN.

SETTING DOUBLE-STRIKE PRINTING

Whenever possible, I prefer to make each of the main modules in a specific program follow a similar pattern. Then, during the coding of the program, I often see similarities emerging that allow coding of "more specific" subroutines that can be called by several modules (or routines). One of the benefits of using subroutines that have very specific functions is that, as the program develops in complexity, it is still possible to keep track of the program flow. Plus, by designing subroutines to do "double duty," as it were, the overall program size doesn't seem to grow geometrically with each added function. If you compare the format of the main parameter setting modules, you should be able to see the similarity in coding.

In the module Set_Enhancement, as in the other main modules (where applicable), we begin by doing two things. First, point [BP] at the correct error message for this module, and second, force the input value to a capital letter. Even though these same two functions are performed in several of the modules and therefore produce redundant code, this is one time when I decided that calling a subroutine to handle these functions was not effective (the exception that proves the rule). Since the error message is different for each of the modules, it would take extra code to keep track of the correct message and this would cause an increase in the necessary code, thereby defeating one of the reasons for using subroutines. The only other preliminary portion of this module is to update the input pointer [DI] so that it points to the VARIABLE (on/off) for this particular request (0 or 1).

SET-PRN makes five enhancement settings available to the user. We need to make sure that the user's input request corresponds to one of these possibilities. By designing the compare (CMP) and jump (J?) set of instructions carefully, it is possible to eliminate more than one possibility with several of the CMP instructions. This allows for faster program execution. In this instance, the

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

user's request is for Double-Strike. The first CMP is for Italics. By using the instructions JA (if above) and JB (if below) after a single compare, we can effectively eliminate half of the other possibilities. Since "D I", use of JB lets us eliminate the possibility of either Underline or Wide enhancements. We are left with only two remaining possibilities rather than the four we would have if we had just used a JE (equal) instruction. (Notice that we have effectively tested for equality without even using the JE instruction. If the input had been "I", program execution would have continued with no jump taken; the old "three birds with two stones trick.")

Again with the next CMP we use both JA and JB, but this time we are also testing for the possibility of an error. If the input is below the value of "E", JB causes the jump to be taken. Otherwise, there is a continuation of code execution with no jump taken. The JA effectively tells us that we have an error, since we know there is no acceptable input allowed with an ASCII value above that of "E". In this case we jump on execution of the JB instruction since our input was "D".

Now that we have arrived at the correct portion of the module for setting the Double-Strike enhancement, the next compare is for equality, since the only remaining options are either correct or incorrect input (error). In an instance where the input is not valid, the program jumps to the error-handling subroutine, an error message is displayed, and program execution ends. But in this case, where the input is correct, a further CMP determines whether the request is to turn the enhancement ON or OFF. The user's request was "0" (ON), so the address of the correct string of information needed by the printer is placed in [BX] and the program jumps to Make_Setting, the subroutine that sends the string to the printer.

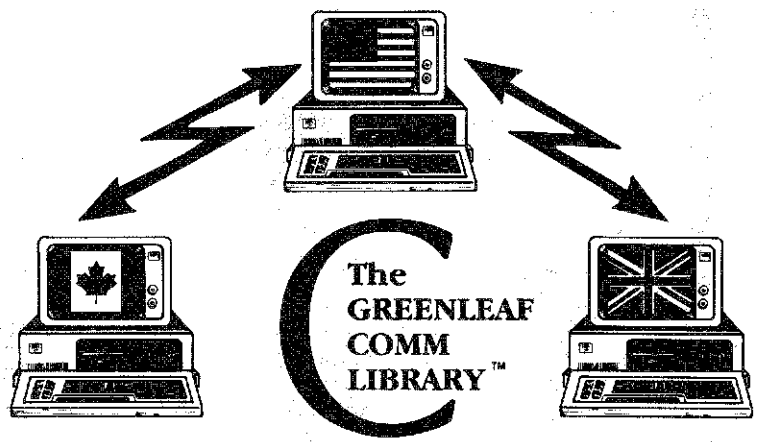
The subroutine Make_Setting is a direct result of designing each of the main modules to follow a similar pattern. When I began coding the program, each module had its own code that sent the information string to the printer. After writing the code for each of the modules, it was apparent that there was a very close similarity between the code sequences being used to end each of the modules. Obviously, program size could be reduced if I made minor code changes to enable the use of a single "module ending" sequence. Make_Setting is the result.

ARE YOU TRYING TO COMMUNICATE ?

C programs can communicate with the world now through the power of **The Greenleaf Comm Library.** Now from the people who brought you **The Greenleaf Functions** General Library for C, comes this rich interrupt driven, ring-buffered asynchronous communications capability.

Over 100 functions in C and assembler to facilitate communications at up to 9600 baud. Up to eight ports at a time. ASCII or XMODEM. X-On/X-Off too. Hayes compatible modems controlled here. Safe too, bet you can't exit your application with interrupts hot. Major applications around the world base their communicating applications on **The Greenleaf Comm Library.** Stop just trying and start really communicating. Get your copy of **The Greenleaf Comm Library** today. For all major C compilers, all models, all versions. For the IBM PC and just about any machine with MSDOS and an 8086. Comes with source code, extensive examples, demo programs, featuring **C-Terminal,** reference card and newsletter. No royalty. $185

Other Products: The Greenleaf Functions General Library, over 220 functions for total control of the IBM PC, with source. $185 for the compilers listed below. (See ordering instructions below).



Specify compiler when ordering: Lattice, Microsoft, Computer Innovations, Mark Williams, or DeSmet. Add $7.00 for UPS Second Day Air (or $5.00 for ground). Texas residents add sales tax. Mastercard, VISA, check, or P.O. In stock, shipped same day.

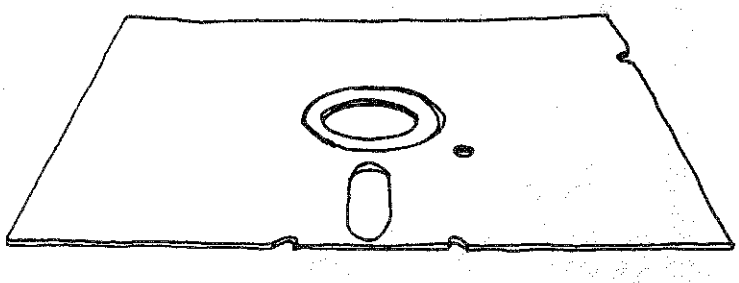
☐ General Libraries	$185
☐ Comm Library	$185
☐ CI186 Compiler	$349
☐ Lattice C	$395
☐ Mark Williams	$475

PRICES ARE SUBJECT TO CHANGE WITHOUT NOTICE.

For Information:
214/446-8641

GREENLEAF SOFTWARE©

2101 HICKORY DR.
CARROLLTON, TX 75006



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

YOU NEED A GOOD LIBRARY

POWER PACKS™

COMPLETE SOURCES
NO ROYALTIES

COMPREHENSIVE C Power Packs include over 1000 functions which provide an integrated environment for developing your applications efficiently. "This is a beautifully documented, incredibly comprehensive set of C Function Libraries."
— Dr. Dobb's Journal, July 1984

USEFUL "...can be used as an excellent learning tool for beginning C Programmers..."
—PC User's Group of Colorado, Jan. 1985

FLEXIBLE Most Compilers and all Memory Models supported.

RECOMMENDED "I have no hesitation in recommending it to any programmer interested in producing more applications code, using more of the PC capabilities, in much less time." — Microsystems, Oct. 1984

- **PACK 1: Building Blocks I** $149
  DOS, Keyboard, File, Printer, Video, Async
- **PACK 2: Database** $399
  B-Tree, Virtual Memory, Lists, Variable Records
- **PACK 3: Communications** $149
  Smartmodem™, Xon/Xoff, X-Modem, Modem-7
- **PACK 4: Building Blocks II** $149
  Dates, Textwindows, Menus, Data Compression, Graphics
- **PACK 5: Mathematics I** $99
  Log, Trig, Random, Std Deviation
- **PACK 6: Utilities I (EXE files)** $99
  Arc, Diff, Replace, Scan, Wipe

Master Card/Visa, $7 Shipping, Mass. Sales Tax 5%
ASK FOR FREE DEMO DISKETTE

NOVUM ORGANUM INC.
SOFTWARE HORIZONS INC.
165 Bedford St., Burlington, MA 01803
(617) 273-4711

If entered at the beginning, the module's function is to call another subroutine, which performs the actual sending of the code to the printer, and then set the Carry Flag to show successful module completion before returning directly to MAIN. If, on the other hand, Make_Setting is entered at the label Error:, the only function is to indicate an error condition before returning to MAIN.

When the program returns to the MAIN code section, after testing for an error condition (which causes an exit), a check is made for more user input on the command line. In this example session we have assumed that the user has only requested that Double-Strike printing be set, so the program will end with a display of the Exit_Message.

Some of you may wonder about the coding of the Ascii_Convert subroutine, in particular, the use of several SHL instructions rather than one MUL to multiply by ten. The reason is quite simply, speed. Figure 1 shows the time needed to complete the multiplication using SHL versus using MUL. In this case space is not at a premium, so the execution speed can be optimized.

FIGURE 1

A comparison of Execution Speed (SHL vs MUL)

code	clocks	code	clocks
mov dx,ax	2	mov dl,0a	4
shl ax,01	2	mul dl	70+
shl ax,01	2		
add dx,ax	3		
shl dx,01	2		
	11	<------ TOTALS ------>	74+

IN SUMMARY

This program is by no means perfect. For instance, it would have been better to use [DX] as the pointer for both the printer code strings and the screen message strings. That way a single module making use of the DOS level function call 40H (write to file or device) could have been used rather than using two DOS function calls: 90H (print string) and 05H (printer output). Nonetheless, I hope that a study of the SET-PRN listing and the reading of this article will help you on the road to improved programming.

PROGRAM AVAILABILITY

I will be happy to supply this program (and accompanying manual with source code) on diskette to any interested parties for a $10.00 handling fee. Versions are available for the following dot-matrix printers: EPSON (MX, RX, FX); PROWRITER (I and II); NEC 8023A-C.

Michael Hoyt
Route 2 Box 65
Solsberry, IN 47459

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ACTIVE TRACE

"Software that lives up to its promises. *When a Basic program doesn't work the way you want it to, this package... will help you track the problem down... Scope is a tool for the beginning, advanced, or professional programmer, and it begins where the cross reference maps leave off."*

Howard Glosser, *Softalk for the IBM Personal Computer*
July '84, pp 120-121

"Extremely useful program... *Anyone doing much programming in Basic should appreciate Active Trace a lot."*

Jerry Pournelle, *Byte Magazine*
April '83, p 234

"A marvelous Basic programming aid... *It's just amazing to watch a program you wrote run under Scope, and debugging becomes if not trivial, then at least doable"*

Thomas Bonoma, *Microcomputing*,
Dec. '83, p 22

"...a really neat utility... *designed to untangle even the most convoluted Basic program....The documentation is almost worth the price of the package."*

Susan Glinert-Cole, *Creative Computing*, July '84, p 210

Active Trace will lead you through your program letting you know variable values (all variables or just those you specify) as they change. Your program's internal activity is presented on your screen, or printer, or it can be saved on disk. **It's simple, effective and works with the BASIC you already own.**

Active Trace **$79.95**
Includes Scope, XREF mapping and documentation

Active Trace is available for most **MS-DOS** and **CPM 2.2** systems and supports the special features of Brand specific versions of Microsoft Basic such as **Basica** on the **IBM-PC**.

AWARECO
Active Software

P.O. Box 695 Gualala, CA 95445
(707) 884-4019
800-358-9120(US) 800-862-4948(CA)

Active trace, Active software, and Scope are trademarks of AWARECO—CPM is a trademark of Digital Research—MS-DOS and Microsoft are trademarks of Microsoft Corporation—IBM-PC is a trademark of IBM Corp.

Periscope ... A Direct Hit!

"A great debugger ... If you've been frustrated by programs that don't behave the same for the debugger as when running alone, or that crash altogether, you should check out Periscope ..."

PC REPORT, Boston Computer Society

"It works, and works well!! In the first day of use I finished up two weeks of problems!! Really excellent!!!"

Peter Loats, Periscope User

Periscope's differences hit home! Its breakout switch gives you control, even when interrupts are disabled. Periscope's unique power helps you solve the difficult problems. Debug device-drivers, memory-resident, and non-DOS programs.

Great for straightening out confused C pointers! Using its breakpoint power, you can stop on reads and writes to ranges of memory. Stop on registers, words, and bytes, too.

It's tough. Periscope's 16K RAM board is write-protected. Runaway programs can't touch crucial debugger code! Examine the system, safely, at any time: Periscope saves the interrupt vectors and buffers it uses, then restores them when done. It uses ROM BIOS calls for functions except file access, so DOS can't clobber itself.

It gets you moving. It's commands are similar to Debug's, so you'll master it quickly. On-line help is there when you need it. You can define the windows you want; you can design easy-to-read memory displays. Periscope supports C, Assembler, BASIC, and Pascal. Comprehensive symbol support gives you a roadmap through memory tying back to your source.

It's risk-free. Periscope is backed up by a 30-day, money-back guarantee! The board is warranted for a year. Technical Support? You get the best there is ... direct from Brett Salter, Periscope's author!

It requires. An IBM PC, XT, AT, Compaq, or close compatible; PC-DOS, 64K RAM; a disk drive and an 80-column monitor. With two monitors, Periscope's screen is displayed on the monitor not in use. With one monitor, a keystroke switches screens.

It's power you can afford. At $295, you can afford Periscope's professional power. The system includes the memory board, breakout switch, debugger software, manual, and quick reference card.

Order now! Call (800) 421-5300, ext. R96, 24 hours a day to order with MasterCard/VISA. Demo disk is $5. Questions? Please call (404) 256-3860.

Get your programs up and running;

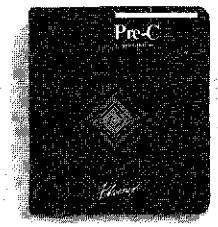
UP PERISCOPE!

Data Base Decisions/14 Bonnie Lane | Atlanta, GA 30328

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programmers' Pfantasies™ by Phoenix.

Phoenix makes programmers' dreams come true. With the best-engineered, highest performance programming tools you can find. A full line of MS-DOS® /PC DOS programs and utilities no other company offers. All designed to help you write, test and deliver the best programs possible. Top-of-the-line quality at a price you can afford.



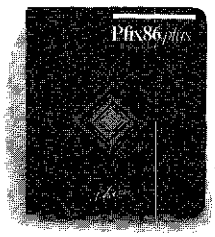
Finally, A Lint For MS-DOS.

Now you can get the full range of features C programmers working in UNIX™ have come to expect from their Lint program analyzer.

With Pre-C™ you can detect structural errors in C programs five times faster than you can with a debugger. Find usage errors almost impossible to detect with a compiler. Cross-check multiple source files and parameters passed to functions. Uncover interface bugs that are difficult to isolate. All in a single pass. Capabilities no C compiler, with or without program analyzing utilities, can offer. In fact, Pre-C outlints Lint, since you can handle analyses incrementally.

Pre-C's flexible library approach lets you maintain continuity across all the programs in your shop, whether you use Pre-C's pre-built libraries, pre-existing functions you already have, or some you might want to buy yourself.

Plus, you're not limited to one particular library, and Pre-C keeps track of all the libraries you're using to make sure that code calls them correctly. **$395.**

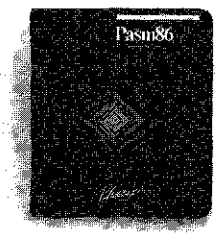


Still Fixing Bugs The Hard Way?

Pfix™86 Plus, the most advanced symbolic debugger on the market, eliminates the endless error searches through piles of listings. Locate instructions and data by symbolic name, using symbolic addresses. Handle larger, overlayed programs with ease.

An adjustable multiple-window display shows source, object code and data, breakpoint settings, current machine register and stack contents simultaneously. An in-line assembler allows program corrections directly in Assembly language. Powerful breakpoint features run a program full speed until a loop has been performed n times.

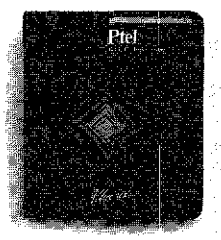
With a single keystroke you can trace an instruction and the action will be immediately reflected in source, object, data, stack, and register windows. Another key begins a special trace mode that executes call and loop instructions at full speed. Designed to work with both Plink™86 and MS® LINK linkage editors. **$395.**



Assemble Programs Twice As Fast.

Pasm™86 will assemble MASM files two to three times faster than MASM 3.0. Pasm86 supports 8086/88, 8087, 80186 and 80286 processors.

With Pasm86's built-in defaults, you can write code quickly since you won't spend hours learning all the control statements needed at the beginning of your program. You can define symbols on the command line. Decide whether you want error messages or not. And, put local symbols within procedures. **$295.**



Get The Lead Out Of Binary File Transfer.

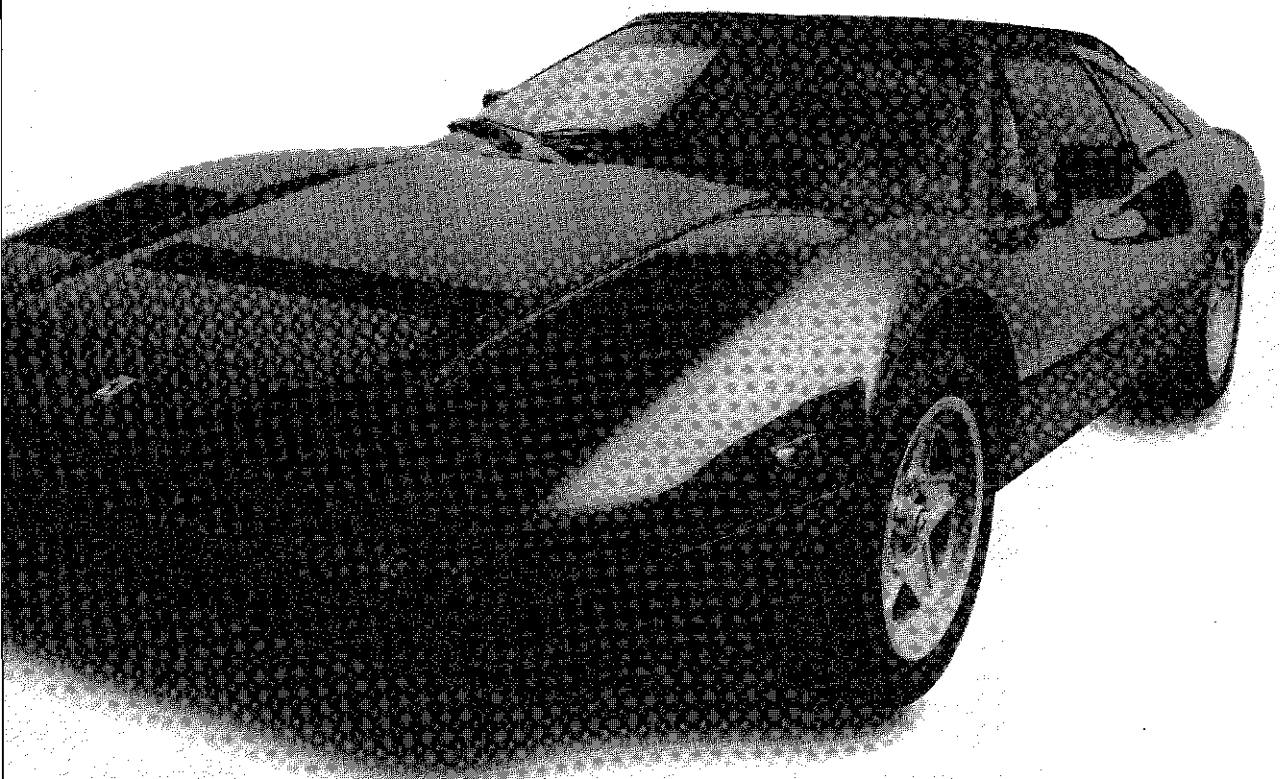
Ptel™ is the universal binary file transfer program for MS-DOS 2.0 or higher. You can move binary files fast and accurately. Upload or download groups of files from Bulletin Boards or remote computers. Move files between dissimilar machines and operating systems. Ptel's advanced binary protocol, Telink,™ offers better-than-Modem7 accuracy and performance. Faster transfer speeds. An on-screen update of error correction, blocks transferred, and transmission time.

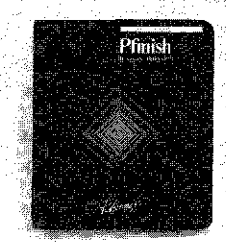
Includes popular Modem7 and XModem protocols. With checksum or CRC. Plus Kermit and ASCII. **$195.**

Special Thanks to Gasten Andrey of Framingham, MA.

Programmer's Pfantasies, Pre-C and Pfinish are trademarks of Phoenix Computer Products Corporation.
MS-DOS and MS are registered trademarks of Microsoft Corporation.
IBM is a registered trademark of International Business Machines Corporation.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

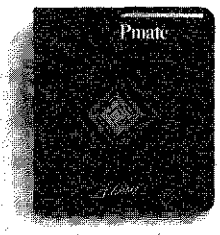




Maximize Your Program's Efficiency.

Pfinish™ delivers the fastest running programs possible. This performance analyzer lets you "zoom in" on the inefficient parts of your program. Whether written in Assembly language. C. PASCAL. FORTRAN. BASIC. Unlike profilers available today, Pfinish understands the structure of your program and reports the amount of activity and time spent in its subroutines or functional groups. Pfinish analyzes both overlaid and memory resident programs. Down to the instruction level. Reports are displayed. Stored on disk. Or printed out. In tabular form or histograms.

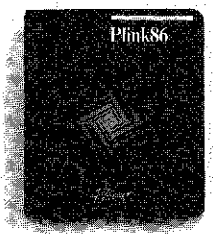
Do a dynamic program scan. Identify the most frequently executed subroutines. Find inefficient code that costs your program valuable time. Rank subroutines by execution frequency. **$395.**



Why Work With A Primitive Editor?

More than a powerful editor, Pmate™ is a text processing language. An emulator of other editors. A language-specific editor for C, PASCAL, and FORTRAN. Pmate can even run in the background!

You get full-screen, single-key editing. Ten editing buffers. Horizontal and vertical scrolling. A "garbage stack" buffer. A built-in macro language with variables, control statements, radix conversion, tracing and 120 commands that you can group and execute with a single keystroke. **$225.**



Why Squeeze Your Program More Than You Have To?

The Plink86 overlay linkage editor brings modular programming to 8086/88-based micros. Write large and complex programs without worrying about memory constraints. Work on modules individually, link them into executable files. Use the same module in different programs. Change the overlay structure of an existing program without recompiling. Use one overlay to access code and data in other overlays. **$395.**

Call (800) 344-7200. In Massachusetts (617) 762-5030.
Or, write.

Phoenix

Phoenix Computer Products Corp.
1420 Providence Highway Suite 115
Norwood, MA 02062

Outside the US, contact: Lifeboat Japan, Tokyo, Japan, Telex #2423296 LBJTYO, Telephone: 03-456-4101 • Repro Haganum (Stevis BV), Dinther, Netherlands, Telex #39581 REHAGNL, Telephone: 01-720-74543, 01-720-75543 • Memory Data, Sundbyberg, Sweden, Telephone: 46-8764-6700 • Roundhill Computer Systems, Marlborough, Wiltshire, UK, Telex #444453 AWARE G, Telephone: 44-467254675.

XT and AT are trademarks of International Business Machines Corporation.
UNIX is a trademark of AT&T Bell Laboratories.
Pasm86, Pfix86 Plus, Ptel, Telink, Pmate and Plink86 are trademarks of Phoenix Software Associates Ltd.
Intel is a registered trademark of Intel Corporation.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
;;;
```

```
page ;*************************************************************************
; If we can't find a SWITCH (the slash character), something is wrong with the
; input syntax, so we'll display an error message and exit.
;*************************************************************************
SLASH:  mov     al,sw                   ;put the switch character into [AL]
        repnz   scasb                   ;scan the string a byte at a time
        jz      PROCESS                 ;jump if "/" is found
        mov     dx,offset Error_2       ;not found, point to error message
        jmp     short DISPLAY_MESSAGE   ;display message and exit

;*************************************************************************
; Once a SWITCH is found, we need to compare the next character against the
; INDICATORs in Table_1 to make sure it's a valid request.
;*************************************************************************
PROCESS:
        push    cx                      ;save the remaining character count
        mov     al,[di]                 ;get indicator into [AL]
        and     al,5f                   ;force to a capital letter
        mov     bx,offset Table_1       ;point to table of valid characters
        call    Check_Character         ;check validity
        jnc     SWITCH                  ;jump if valid
        mov     dx,offset Error_3       ;point to error message
        jmp     short DISPLAY_MESSAGE   ;display message and exit

;*************************************************************************
; If the SWITCH is valid, we use the SWITCH character to find the correct code
; module to set this particular switch request.
;*************************************************************************
SWITCH: sub     al,41                   ;use the input as an offset pointer
        shl     al,01                   ;multiply by 2 for correct offset
        mov     bx,offset Table_2       ;point to address table
        xor     ah,ah                   ;clear upper half of [AX]
        add     bx,ax                   ;point to address of correct procedure
        inc     di                      ;point to the user's choice for
                                        ;for this particular switch indicator
        call    word ptr [bx]           ;call correct procedure

page ;*************************************************************************
; If everything has gone ok in the module we just called, we check for the end
; of the command line and if we haven't reached it, loop through the MAIN
; procedure again.
;*************************************************************************
        jc      DISPLAY_MESSAGE         ;error ?  display message and exit
        inc     di                      ;point to the next input character
        cmp     byte ptr [di],cr        ;are we finish ? (is it a CR ?)
        mov     dx,offset exit_message  ;point to the standard exit message
        je      DISPLAY_MESSAGE         ;yes, we are done
        pop     cx                      ;recover the count of remaining charctr
        jmp     short SLASH             ;do next switch setting

;*************************************************************************
; If we reach this portion of the MAIN code, we are either finished or an error
; has been encountered .
;*************************************************************************
DISPLAY_MESSAGE:
        mov     ah,09                   ;use DOS function call
        int     21                      ;[DX] points to display string
        mov     bx,offset Cr_Lf$        ;use CR-LF to clear printer buffer
        call    Print_String            ;send to printer
        mov     ah,4c                   ;terminate this process (program)
        int     21                      ;and return control to the
MAIN            endp                    ;"parent process"  -  in this case DOS
```

```
page ;======================  CALLED MODULES  ==============================
;*************************************************************************
; This module is used to set the printer to the Pitch requested by the user.
; Once that Pitch is determined, the codes needed to set that particular Pitch
; are recovered from look-up Table_3.
;
;       INPUT   -       [DI] points to user's choice of settings
;
;       OUTPUT  -       carry flag is set to:
;                               0 = valid request
;                               1 = not valid request
;
;       CHANGED -       [AX], [BX], [CX], [DX], [DI], [BP]
;*************************************************************************
Set_Pitch       proc near
        mov     bp,offset Error_4       ;point to the proper error message
        mov     al,[di]                 ;get choice into [AL]
        and     al,5f                   ;force to a capital letter

Pica:
        cmp     al,'N'                  ;is it pica ?
        jb      Elite                   ;jump if less than
        ja      Proportional            ;jump if greater than
        mov     bx,offset Pica$         ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

Elite:
        cmp     al,'E'                  ;is it elite ?
        jne     Compressed              ;no, see if it's compressed
        mov     bx,offset Elite$        ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

Compressed:
        cmp     al,'C'                  ;is it compressed ?
        jne     Pitch_Error             ;no, it's not a valid request
        mov     bx,offset Compressed$   ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

Proportional:
        cmp     al,'P'                  ;is it proportional ?
        jne     Pitch_Error             ;no, it's not a valid request
        mov     bx,offset Proportional$ ;point to the correct string
        jmp     Make_Setting            ;send the code string to the printer

Pitch_Error:
        jmp     Error                   ;indicate error and return
Set_Pitch       endp
```

PROTECTED 2004 BY UNZIP.RU
ELECTRONIC REPRODUCTION PROHIBITED

```
page ;*****************************************************************************
; This module to set the user's requested enhancement uses a specially
; designed compare (CPM) and jump (JMP) set of instructions to eliminate half
; of the remaining possiblilities with each compare instruction.  It can be
; diagrammed as follows.
;                                       Italics
;                                      /   |   \
;                                     /    |    \
;                                Enhanced  |  Underline
;                               /     \    |    /     \
;                              /       \   |   /       \
;                     Double-Strike     \  |  /       Wide
;                              \       \  \ | /       /
;                               \       \  \|/       /
;                                (  OPPS!!!  AN ERROR!!!  )
;
; As in the other modules, the code string to be sent to the printer is
; gotten from look-up table_3.
;
;       INPUT   -       [DI] points to user's choice of settings
;
;       OUTPUT  -       carry flag is set to:
;                               0 = valid request
;                               1 = not valid request
;
;       CHANGED -       [AX], [BX], [CX], [DX], [DI], [BP]
;*****************************************************************************
Set_Enhancement proc near
        mov     bp,offset Error_5       ;point to error message
        mov     al,[di]                 ;get choice into [AL]
        and     al,5f                   ;force to a capital letter
        inc     di                      ;point to next character of input
;-----------------------------------------------------------------------------
Italics:
        cmp     al,'I'                  ;is it italics ?
        jb      Enhanced                ;jump if less than
        ja      Underline               ;jump if greater than
        cmp     byte ptr [di],'0'       ;is indicator set for ON or OFF ?
        jne     Italics_Off             ;indicator <> 0, turn italics OFF ?
        mov     bx,offset Italics_On$   ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

Italics_Off:
        cmp     byte ptr [di],'1'       ;does indicator = 1 ?
        jne     Enhancement_Error       ;no, so indicate error
        mov     bx,offset Italics_Off$  ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

page ;-----------------------------------------------------------------------------

Enhanced:
        cmp     al,'E'                  ;is it enhanced ?
        jb      Double_Strike           ;jump if less than
        ja      Enhancement_Error       ;jump if greater than
        cmp     byte ptr [di],'0'       ;is indicator set for ON or OFF ?
        jne     Enhanced_Off            ;indicator <> 0, turn enhanced OFF ?
        mov     BX,offset Enhanced_On$  ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

Enhanced_Off:
        cmp     byte ptr [di],'1'       ;does indicator = 1 ?
        jne     Enhancement_Error       ;no, so indicate error
        mov     bx,offset Enhanced_Off$ ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer
;-----------------------------------------------------------------------------

Double_Strike:
        cmp     al,'D'                  ;is it double strike ?
        jne     Enhancement_Error       ;no, it's not a valid request
        cmp     byte ptr [di],'0'       ;is indicator set for ON or OFF ?
        jne     Double_Strike_Off       ;indicator <> 0, double strike off?
        mov     bx,offset Double_Strike_On$     ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

Double_Strike_Off:
        cmp     byte ptr [di],'1'       ;does indicator = 1 ?
        jne     Enhancement_Error       ;no, so indicate error
        mov     bx,offset double_Strike_Off$    ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

page ;-----------------------------------------------------------------------------

Underline:
        cmp     al,'U'                  ;is it underline ?
        ja      Wide                    ;jump if greater than
        jb      Enhancement_Error       ;no, it's not a valed request
        cmp     byte ptr [di],'0'       ;is indicator set for ON or OFF ?
        jne     Underline_Off           ;indicator <> 0, turn underline off?
        mov     bx,offset Underline_On$ ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

Underline_Off:
        cmp     byte ptr [di],'1'       ;does indicator = 1 ?
        jne     Enhancement_Error       ;no, it's not a valid request
        mov     bx,offset Underline_Off$        ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

;-----------------------------------------------------------------------------

Wide:
        cmp     al,'W'                  ;is it double width ?
        jne     Enhancement_Error       ;no, it's not a valid request
        cmp     byte ptr [di],'0'       ;is indicator set ON or OFF ?
        jne     Wide_Off                ;indicator <> 0, turn dbl width off
        mov     bx,offset Wide_On$      ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

Wide_Off:
        cmp     byte ptr [di],'1'       ;does indicator = 1 ?
        jne     Enhancement_Error       ;no, so indicate error
        mov     bx,offset Wide_Off$     ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

Enhancement_Error:
        jmp     Error                   ;indicate error and return
Set_Enhancement endp
```

ELECTRONIC REPRODUCTION PROHIBITED PRODUCED 2004 BY UNZ.ORG

```
page ;*****************************************************************
; The module that sets the printer's line spacing increment must be able to
; handle the input of numbers when the user request a non-standard line space.
; Numbers will be handled by 1st calling a subroutine (Number_Check) to make
; sure that numbers have actually been input and then calling the subroutine
; Ascii_Convert to change the 2 bytes that have been input to a single
; Hexadecimal number that can then be sent to the printer.  If the user inputs
; more than 2 numbers, only the 1st 2 are used.  Any remaining numbers will be
; ignored.
;
;       INPUT   -       [DI] points to user's choice of settings
;
;       OUTPUT  -       carry flag is set to:
;                               0 = valid request
;                               1 = not valid request
;
;       CHANGED -       [AX], [BX], [CX], [DX], [DI], [BP]
;*****************************************************************
Set_Spacing     proc near
        mov     bp,offset Error_6       ;point to error message
        cmp     byte ptr [di],3a        ;could it be variablle spacing ?
        jb      Variable_Spacing        ;maybe, so handle a special way
        mov     al,[di]                 ;no, so get choice into [AL]
        and     al,5f                   ;force to a capital letter

 One_Sixth:
        cmp     al,'N'                  ;is it 1/6" spacing ?
        jne     One_Eighth              ;no, see if it's 1/8"
        mov     bx,offset Spacing_6$    ;point to correct string
        jmp     short Spacing_Set       ;set spacing and return

 One_Eighth:
        cmp     al,'A'                  ;is it 1/8" spacing ?
        jne     Spacing_Error           ;no, it's not a valid request
        mov     bx,offset Spacing_8$    ;point to correct string

 Spacing_Set:
        call    Print_String            ;set printer to 1/8" SPACING
        jmp     No_Error                ;set the carry flag and return

 Variable_Spacing:
        mov     dx,offset Variable_Spacing$     ;point to correct string
        call    Print_String            ;send part of code for variable spacin
        jmp     Number_Handler          ;handle the numeric input

 Spacing_Error:
        jmp     Error                   ;set the carrry flag and return
Set_Spacing     endp

page ;*****************************************************************
; The module that sets the printer's left margin, like the module to set the
; line spacing, must be able to handle the input of numbers.  Numbers will be
; handled by 1st calling a subroutine (Number_Check) to make sure that numbers
; have actually been input and then calling the subroutine Ascii_Convert to
; change the 2 bytes that have been input to a single Hexadecimal number that
; can then be sent to the printer.  If the user inputs more than 2 numbers,
; only the 1st 2 are used.  Any remaining numbers will be ignored.
;
;       INPUT   -       [DI] points to user's choice of settings
;
;       OUTPUT  -       carry flag is set to:
;                               0 = valid request
;                               1 = not valid request
;
;       CHANGED -       [AX], [BX], [CX], [DX], [DI], [BP]
;*****************************************************************
Set_Margin      proc near
        mov     bp,offset Error_7       ;point to error message
        mov     bx,offset Margin$       ;point to correct string
        call    Print_String            ;send part of code to set margin
        jmp     Number_Handler          ;handle the numeric input
Set_Margin      endp

page ;*****************************************************************
; This is the simplest module since we only have to determine a choice between
; 2 settings.  When the choice is made, the string that is to be sent to the
; printer comes from look-up Table_3.
;
;       INPUT   -       [DI] points to user's choice of settings
;
;       OUTPUT  -       carry flag is set to:
;                               0 = valid request
;                               1 = not valid request
;
;       CHANGED -       [AX], [BX], [CX], [DX], [DI], [BP]
;*****************************************************************
Set_Direction   proc near
        mov     bp,offset Error_8       ;point to error message
        mov     al,[di]                 ;get choice into [AL]
        and     al,5f                   ;force to a capital letter

Unidirectional:
        cmp     al,'U'                  ;is it unidirectional ?
        jne     Bidirectional           ;no, see if it's bidirectional
        mov     bx,offset Unidirectional$       ;point to correct string
        jmp     Make_Setting            ;set print mode and return

Bidirectional:
        cmp     al,'B'                  ;is it bidirectional ?
        jne     Direction_Error         ;no, it's not a valid request
        mov     bx,offset Bidirectional$        ;point to correct string
        jmp     Make_Setting            ;send the code string to the printer

Direction_Error:
        jmp     Error                   ;indicate error and return
Set_Direction   endp

page ;*****************************************************************
; After determining whether to set vertical or horizontal tab spacing, we once
; again use the combination of checking the numerical input and then converting
; it to a single hexadecimal number for sending to the printer.
;
;       INPUT   -       [DI] points to user's choice of settings
;
;       OUTPUT  -       carry flag is set to:
;                               0 = valid request
;                               1 = not valid request
;
;       CHANGED -       [AX], [BX], [CX], [DX], [DI], [BP]
;*****************************************************************
Set_Tabs        proc near
        mov     bp,offset Error_9       ;point to error message
        mov     al,[di]                 ;get the choice into [AL]
        and     al,5f                   ;force to a capital letter

H_Tab:  cmp     al,'H'                  ;is it a request for horizontal tabs ?
        jne     V_Tab                   ;no, see if it's for veritical tabs
        mov     bx,offset Horizontal_Tab$       ;yes, point to correct string
        jmp     short Tabs_Set          ;set the horizontal tabs and return

V_Tab:  cmp     al,'V'                  ;is it a request for vertical tabs ?
        jne     Tab_Error               ;no, it's not a valid request
        mov     bx,offset Vertical_Tab$ ;yes, so point to the correct string
```

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
Tabs_Set:
        call    Print_String            ;send the initial string to the printer
        inc     di                      ;point to the next input character
        jmp     Number_Handler          ;handle the numeric input

Tab_Error:
        jmp     Error                   ;set the carry flag and return
Set_Tabs        endp

page ;*************************************************************************
; This module sets both the Length of the form and the Number of lines to be
; skipped at the perforation.  If the User includes the 'D', default settings
; of 66 lines (form lenght) and 6 lines (perforation skip) are made.  This
; results in 60 usable lines per page.
;
;       INPUT   -       [DI] points to user's choice of settings
;
;       OUTPUT  -       carry flag is set to:
;                               0 = valid request
;                               1 = not valid request
;
;       CHANGED -       [AX], [BX], [CX], [DX], [DI], [BP]
;*************************************************************************
Set_Forms       proc near               ;set Form Length and Perforation Skip
        mov     bp,offset Error_10      ;point to error message
        mov     al,[di]                 ;get choice into [AL]
        and     al,5f                   ;force to a capital letter

Form:
        cmp     al,'F'                  ;is it set the Form Length ?
        ja      Skip_Lines              ;no, see if it's set Perforation Skip
        jb      Default                 ;no, see if it's set "defaults"
        mov     bx,offset Form_Length$  ;yes, so point to the correct string
        jmp     short Form_Set          ;set spacing and return

Default:
        cmp     al,'D'                  ;is it Form Length ?
        jne     Form_Error              ;no, it's not a valid request
        mov     bx,offset Default$      ;yes, so point to the correct string
        call    Print_String            ;set default settings
        jmp     No_Error                ;done, so return

Skip_Lines:
        cmp     al,'S'                  ;is it Perforation Skip ?
        jne     Form_Error              ;no, it's not a valid request
        mov     bx,offset Skip_Line$    ;yes, so point to the correct string

Form_Set:
        call    Print_String            ;send the initial string to the printe
        inc     di                      ;point to the next input character
        jmp     Number_Handler          ;handle the numeric input

Form_Error:
        jmp     Error                   ;set the carry flag and return
Set_Forms       endp

page ;*************************************************************************
; This subroutine simply compares the input character against the valid
; SWITCHes contained in Table_1.
;
;       INPUT   -       [BX] points to Table_1
;                       [AL] contains the character that was input
;
;       OUTPUT  -       carry flag is set to:
;                               0 = valid request
;                               1 = not valid request
;
;       CHANGED -       [BX], [CX]
;*************************************************************************
Check_Character proc near
        xor     cx,cx                   ;clear [CX] to use as counter
        mov     cl,count                ;set [CX] to # of available switches
        cmp     al,[bx]                 ;do we have a valid character
        je      Return                  ;yes, so return
        inc     bx                      ;point to next valid character
        loop    Check_Character         ;loop until all are checked
        stc                             ;set carry flag = 1, not valid
        ret                             ;done, so return
Return: clc                             ;set carry flag = 0, valid
        ret                             ;done, so return
Check_Character endp

;*************************************************************************
; Any time a parameter setting requires numerical input, this subroutine is
; called.
;
;       INPUT   -       [DI] points to the input character
;                            hopefully it is the 1st of 2 numbers
;                       [BP] points to the correct error message
;                            for this SWITCH setting
;
;       OUTPUT  -       carry flag is set to:
;                               0 = valid request
;
;       CHANGED -       [AX], [DI]
;*************************************************************************
Number_Handler  proc near
        call    Number_check            ;see if we have numeric input
        inc     di                      ;point to next input character
        call    Number_Check            ;see if input is also a number
        dec     di                      ;point to 1st of two numbers
        call    Ascii_Convert           ;convert 2-byte string to hexadecimal
        call    Print                   ;send to the printer
        jmp     short No_Error          ;indicate no error and return
Number_Handler  endp
page ;*************************************************************************
; This subroutine is only called by the Number_Handler.  It simply determines
; if the input is in the range of 0 to 9 (a decimal number).
;
;       INPUT   -       [DI] points to the input character
;                            hopefully it is the 1st of 2 numbers
;                       [BP] points to the correct error message
;                            for this SWITCH setting
;
;       OUTPUT  -       carry flag is set to:
;                               1 = not valid request
;*************************************************************************
Number_Check    proc near               ;on entry [DI] points to input char
        cmp     byte ptr [di],'0'       ;is it less than 0 ?
        jb      Error                   ;not a number, not valid request
        cmp     byte ptr [di],'9'       ;is it greater than 9 ?
        ja      Error                   ;yes, not a valid request
        ret                             ;done, (a valid number) so return
Number_check    endp
```

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
page ;*************************************************************************
; This procedure converts a 2-byte string of numbers (ASCII input) to a single
; byte of hexadecimal information that can then be sent to the printer.
;
;       INPUT   -       [DI] points to the 1st of the 2-bytes of numeric
;                       input
;
;       OUTPUT  -       [DL] contains the hexadecimal value for the printer
;
;       CHANGED -       [AX], [DI], FLAGS
;
; <NOTE>        When this subroutine is called, we have assured that we have 2
; decimal numbers to convert.  Rather than moving values on and off the stack
; using PUSHes and POPs, we will use registers for temporary storage, since
; that is much faster.
;*************************************************************************
Ascii_Convert   proc near               ;on entry [DI] point to 2-byte string
        xor     ax,ax                   ;clear [AX] to receive ASCII number
        mov     al,[di]                 ;get 1st ASCII digit
        sub     al,30                   ;convert to hexadecimal
        mov     dx,ax                   ;save hex result in [DX]
        shl     ax,01                   ;multiply by 2
        shl     ax,01                   ;multiply by 4
        add     dx,ax                   ;add 2 values (multiply by 5)
        shl     dx,01                   ;multiply by 10
        inc     di                      ;point to 2nd ASCII digit
        mov     al,[di]                 ;get 2nd digit into [AL]
        sub     al,30                   ;convert to hexadecimal
        add     dl,al                   ;add to 1st value
        ret                             ;done, so return
Ascii_Convert   endp                    ;[BX] = hex value of 2-byte string

page ;*************************************************************************
; This subroutine is used for 2 functions.  First, it handles the sending of
; the code strings to the printer.  Second, it takes care of setting the carry
; flag to indicate error conditions.

        INPUT   -       [BX] points to the code string to be sent
                        [CX] contains the length of the string
                        [BP] points to the appropriate error message

        OUTPUT  -       carry flag is set to:
                                0 = valid request
                                1 = not valid request
                      *  [DX] points to the error message
;
; <NOTE>        If the subroutine is used only for setting the carry flag, it
; is entered at either No_Error or Error, in which case the error message
; pointer is moved to [DX] before exiting the procedure.
;*************************************************************************
Make_Setting    proc near
        call    Print_String            ;send the string to the printer

No_Error:
        clc                             ;set flag = 0, valid
        ret                             ;done, so return

Error:  mov     dx,bp                   ;recover the pointer to the correct
                                        ;error message
        stc                             ;set flag = 1, not valid
        ret                             ;done, so return
Make_Setting    endp
```

```
page ;*************************************************************************
; This subroutine sends the code string to the printer.
;
;       INPUT   -       [BX] points to the code string
;
;       OUTPUT  -       code is sent to the printer
;                       no is information returned
;
;       CHANGED -       [AX], [BX], [CX], [DX]
;*************************************************************************
Print_String    proc near               ;on entry [BX] points to string
        xor     cx,cx                   ;clear [CX] for use as a counter
        mov     cl,[bx]                 ;get count of bytes to be sent
Next:   inc     bx                      ;point to 1st code byte
        mov     dl,[bx]                 ;get byte into [DL]
        call    Print                   ;send to printer
        loop    Next                    ;loop until entire string is sent
        ret                             ;done, so return
Print_String    endp


;*************************************************************************
; This subroutine sends the information contained in [DL] to the printer.
;
;       INPUT   -       [DL] contains character to be sent to printer
;
;       OUTPUT  -       a byte is sent to printer 1 [PRN] (Lpt1:)
;                       no information is return
;
;       CHANGED -       [AX]
;*************************************************************************
Print           proc near               ;on entry [DL] has character
        mov     ah,05                   ;indicate print contents of [DL]
        int     21                      ;use DOS function call
        ret                             ;done, so return
Print           endp
```

PRODUCED BY WWW.BBNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
page ;======================  LOOK-UP TABLES  ================================

Table_1 db      'DEFLMPST'              ;valid switch indicators
count   equ     $-Table_1               ;number of switches
        db      20 dup(?)               ;save room for more switches

;*****************************************************************************
; This table contains the addresses of each of the modules that handle the
; SWITCH settings.  By designing the table so that each letter of the alphabet
; is assigned 2 bytes, we can easily perform the necessary calculations to find
; the correct offset into the table that has the address of the particular
; module that we want.
;*****************************************************************************
Table_2:
        dw 0000                         ;A
        dw 0000                         ;B
        dw 0000                         ;C
        dw Set_Direction                ;D - Set Print Direction
        dw Set_Enhancement              ;E - Set Character Enhancements
        dw Set_Forms                    ;F - Set Form-page/form length
        dw 0000                         ;G
        dw 0000                         ;H
        dw 0000                         ;I
        dw 0000                         ;J
        dw 0000                         ;K
        dw Set_Margin                   ;L - Set Left Margin
        dw 0000                         ;M
        dw 0000                         ;N
        dw 0000                         ;O
        dw Set_Pitch                    ;P - Set Pitch
        dw 0000                         ;Q
        dw 0000                         ;R
        dw Set_Spacing                  ;S - Set Line Spacing
        dw Set_tabs                     ;T - Set Tabs -- Horizontal or Vertical
        dw 0000                         ;U
        dw 0000                         ;V
        dw 0000                         ;W
        dw 0000                         ;X
        dw 0000                         ;Y
        dw 0000                         ;Z

page ;*****************************************************************************
; This table contains the code strings that need to be sent to the printer for
; each of the desired parameters.  Room is saved for longer strings so that
; users may add extra code if desired.  Notice that the 1st byte of each string
; is the count of characters to be sent to the printer.
;*****************************************************************************
Table_3:
Pica$                   db 06,1b,50,12,1b,70,00,00,00   ;the PICA code cancels
Compressed$             db 01,0f,00,00,00,00,00,00,00   ;both compressed print
Proportional$           db 03,1b,70,01,00,00,00,00,00   ;and proportional print
Elite$                  db 02,1b,4d,00,00,00,00,00,00
Double_Strike_On$       db 02,1b,47,00,00,00,00,00,00
Double_Strike_Off$      db 02,1b,48,00,00,00,00,00,00
Enhanced_On$            db 02,1b,45,00,00,00,00,00,00
Enhanced_Off$           db 02,1b,46,00,00,00,00,00,00
Wide_On$                db 03,1b,57,01,00,00,00,00,00
Wide_Off$               db 03,1b,57,00,00,00,00,00,00
Italics_On$             db 02,1b,34,00,00,00,00,00,00
Italics_Off$            db 02,1b,35,00,00,00,00,00,00
Underline_On$           db 03,1b,2d,01,00,00,00,00,00
Underline_Off$          db 03,1b,2d,00,00,00,00,00,00
Spacing_6$              db 02,1b,32,00,00,00,00,00,00
Spacing_8$              db 02,1b,30,00,00,00,00,00,00
Variable_Spacing$       db 02,1b,41,00,00,00,00,00,00   ;xx/72"
Margin$                 db 02,1b,6c,00,00,00,00,00,00
Unidirectional$         db 03,1b,55,01,00,00,00,00,00
Bidirectional$          db 03,1b,55,00,00,00,00,00,00
Horizontal_Tab$         db 03,1b,65,00,00,00,00,00,00
Vertical_Tab$           db 03,1b,65,01,00,00,00,00,00
Default$                db 06,1b,43,42,1b,4e,06,00,00   ;form=66,skip=6
Form_Length$            db 02,1b,43,00,00,00,00,00,00
Skip_Line$              db 02,1b,4e,00,00,00,00,00,00
Cr_Lf$                  db 02,0a,0d,00,00,00,00,00,00   ;CR/LF sequence

page ;======================  MESSAGE STRINGS  ===============================

Error_1         db 'Please specify at least one(1) SWITCH...$'
Error_2         db 'SWITCH must be proceded by "/"...$'
Error_3         db 'Invalid SWITCH included...$'
Error_4         db 'Invalid INDICATOR for character pitch...$'
Error_5         db 'Invalid INDICATOR for enhancement or missing VARIABLE...$'
Error_6         db 'Invalid INDICATOR for line spacing or missing VARIABLE...$'
Error_7         db 'Invalid VARIABLE for left margin...$'
Error_8         db 'Invalid INDICATOR for printer direction request...$'
Error_9         db 'Invalid VARIABLE for horizontal tab...$'
Error_10        db 'Invalid VARIABLE for form length...$'
Exit_message    db 'The printer has been set to the requested parameters...$'

Copyright       db 'command line SET_PRINTER UTILITY version 1.0 '
                db '(c) Copyright by Michael L Hoyt, 1984 '
                db 'All rights reserved. '
                db 'EPSON Version '
CODE    ends
END     MAIN
```

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

REPOSITIONING THE STARS IN STARFINDER

by Melvin O. Duke

Starfinder ON DISPLAY is a new, User-Supported program for IBM PCs (including the PCjr), which uses the screen of the computer as a "Window in the Sky." It permits a user to view the stars in any portion of the sky, and to change the view according to the following:

Find a star by its common name.
Find a star by its scientific name.
Find a constellation by its name.
Print the common names of the stars.
Print the scientific names of the stars.
Move the stars farther apart.
Move the stars closer together.
Move the stars to the left.
Move the stars to the right.
Move the stars up.
Move the stars down.
Rotate the stars clockwise.
Rotate the stars counterclockwise.

This article describes the methods and techniques used to display the stars and to reposition them according to the wishes of the viewer.

When Starfinder ON DISPLAY begins, an initial view of the sky is shown (including the stars in the Constellation Orion). For this view, and for all subsequent views, the stars appear as they would in the early evening, with the brightest stars appearing first, followed by less bright stars. The dimmer stars even twinkle.

After completing each view, the user chooses his next view by selecting another action from the list given above. The new positions of all stars (including those not shown) are recalculated as the next view appears.

THE FIRST POSITIONS

Before the stars can be repositioned, they must be located the first time. If one looks at startables (in books, almanacs, or other star publications), one quickly finds that the positions of the stars are given in terms of right-ascension and declination. It is necessary to convert from those terms to others that can be used in plotting on a computer screen and in repositioning for other views.

In addition, the startables may give a distance to each star. This value is not necessarily helpful when one is trying to show the appearance of the sky from earth although it would be useful if one were somewhere else in the universe.

One simplifying assumption is necessary. We will assume that all stars are the same distance from earth and that the distance is 1. With that assumption, coordinates of each of the stars (in terms of x, y, and z) can be calculated, using the right-ascension and declination values shown in the startables.

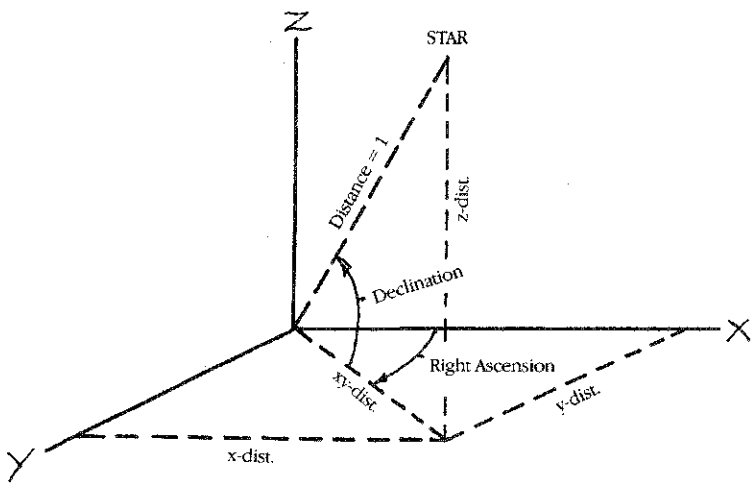


Figure 1. Obtaining x, y, and z-Coordinates.

Based on Figure 1, the z-coordinate of a star can be calculated, using an arc-sine function, with a distance of 1, and the angle which is the same as the declination. A distance from the center in the xy-plane can also be calculated, using the arc-cosine function (or the square root of 1 minus the z-distance squared). This distance in the xy-plane can then be used to determine both the x-coordinate and the y-coordinate.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Advanced Screen Management made easy

Now a professional software tool from Creative Solutions.

WINDOWS FOR C™

More than a window display system, WINDOWS FOR C is a video tool kit for all screen management tasks.

- **Pop-up menus and help files**
- **Unlimited files and windows**
- **Instant screen changes**
- **Complete color control**
- **Horizontal and vertical scrolling**
- **Word wrap**
- **Highlighting**
- **Auto memory management**
- **Plus a library of over 50 building block subroutines**

Designed for portability. Easy to learn, easy to use.

Once you've tried WINDOWS FOR C, you'll wonder how you ever managed without it.

Full support for IBM PC/XT/AT and compatibles, plus interfaces for non-IBM computers; Lattice C, CI-C86, Mark Wm. C, Aztec C, Microsoft C, DeSmet C (PC/MSDOS),

NEW Ver. 3.1
Enhanced portability.
Topview compatible.
WINDOWS FOR C $195
(specify compiler & version)
Demo disk and manual $ 30
(applies toward purchase)
Full source available.
No royalties.

Creative Solutions
21 Elm Ave., Box T7,
Richford, VT 05476
802-848-7738
Master Card & Visa Accepted
Shipping $2.50
VT residents add 4% tax.

Before the x-coordinate and the y-coordinate can be found, we need to convert the right-ascension value from its 24-hour measure (hours, minutes, seconds) to a degree measure, based upon 360 degrees around the horizontal plane. With this degree measure, it is then simple to convert the xy-distance to an x-coordinate and a y-coordinate, using the arc-sine and arc-cosine functions.

Starfinder ON DISPLAY performs these calculations for each star and plots each star within the "Window in the Sky", before calculating the position of the next star.

WINDOW IN THE SKY

In order to have a reasonable view of a portion of the sky, it was necessary to select a "window", based upon an initial guess, making adjustments for the shape of the screen and the fact that information can appear on 25 lines, in 80 character positions across the screen.

The x-coordinates will appear in horizontal positions across the screen, while the z-coordinates will appear in vertical positions on the screen. Y-coordinates which are positive (in front of the viewer) will permit the display of a star, while y-coordinates which are negative (behind the viewer) will prohibit the display of a star.

The coordinates finally chosen, through some trial and error, were x-coordinates between -.32 and +.32, and z-coordinates between -.20 and +.20. Thus, any star whose coordinates are in accord with these definitions will be displayed, while all others will not.

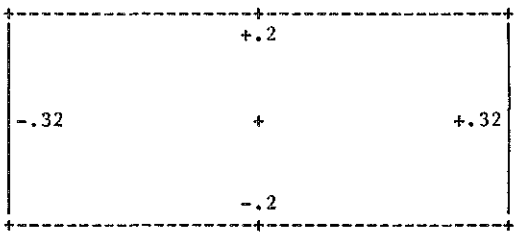


Figure 2. Window in the Sky.

For a better view of the sky, four separate characters were chosen to represent any specific star, based upon the brightness (visual magnitude) of the star.

REPOSITIONING THE STARS

Repositioning the stars is done with the use of what are called "Array Transform Functions." Before these functions can be used, however, the x, y, z-coordinates have to be changed to

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

row vectors. This is done simply by putting a 1 before each set of coordinates, as:

Direction:	x	y	z
Coordinates:	.300	.400	.867
Row Vector:	1	.300	.400	.867

The principal array transform function used in Starfinder ON DISPLAY is the ROTATE function. It is used for rotating points about the x-axis, y-axis, or z-axis.

Moving the stars up or down is simply a rotation about the x-axis. Moving the stars to the left or right is simply a rotation about the z-axis. Rotating the stars clockwise or counter clockwise is a rotation about the y-axis.

The following example illustrates moving the stars to the left, which is a rotation about the z-axis.

The array transform function for rotating left, about the z-axis is defined as:

1	0	0	0
0	cos(t)	sin(t)	0
0	-sin(t)	cos(t)	0
0	0	0	1

Figure 3. Array Transform for Rotation about the z-axis.

When a row vector is multiplied by this array transform function, the resulting row vector contains the representation of the new point, as:

1	x	y	z	*	1	0	0	0	=	1	x'	y'	z'
					0	c	s	0					
					0	-s	c	0					
					0	0	0	1					

For example, if a user chooses to move the stars to the left 30 degrees, the array transform function would be:

1	0	0	0
0	.866	.500	0
0	-.500	.866	0
0	0	0	1

C + C UTILITY LIBRARY = PRODUCT

• We have over 200 complete, tested, and, documented functions. All source code and demo programs are included.

• The library was specifically designed for software development on the IBM PC, XT, AT and compatibles. There are no royalties.

• Over 95% of the source code is written in C. Experienced programmers can easily "customize" functions. Novices can learn from the thorough comments.

We already have the functions you are about to write

Concentrate on software development—not writing functions.

THE C UTILITY LIBRARY includes:

• Best Screen Handling Available • Windows • Full Set of Color Graphics Functions • Better String Handling Than Basic • DOS Directory and File Management • Execute Programs, DOS Commands and Batch Files • Complete Keyboard Control • Extensive Time/Date Processing • Polled ASYNC Communications • General DOS/BIOS gate • And More •

• The Library is compatible with: Lattice, Microsoft, Computer Innovations, Mark Williams and DeSmet.

C Compilers: Lattice C—$349, Computer Innovations C86—$329; Mark Williams C—$449.

C UTILITY LIBRARY $149

Order direct or through your dealer. *Specify compiler* when ordering. Add $4.00 shipping for UPS ground, $7.00 for UPS 2-day service. NJ residents add 6% sales tax. Master Card, Visa, check or P.O.

ESI ESSENTIAL SOFTWARE, INC
P.O. Box 1003 Maplewood, New Jersey 07040 914/762-6605

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Multiplying an initial point (which had been changed to a row vector) of

	x	y	z
1	.300	.400	.867

by the array transform function would result in a new row vector of

	x'	y'	z'
1	.060	.496	.867

The new x', y' and z' coordinates can then be tested to see whether or not they fit within the "Window in the Sky" (in this case they do not) and plotted, as required.

Moving the stars farther apart or moving them closer together is done with another array transform function, known as the scaling function. It may be represented as:

1	0	0	0
0	scale	0	0
0	0	scale	0
0	0	0	scale

Figure 4. Array Transform for Scaling.

The scaling transformation is used in the same way as the rotation transformation; in that the row vector that represents each point is multiplied by the scaling transformation, giving a new row vector, which contains the new coordinate values.

For example, if a user wanted to move the stars closer together and chose a scale factor of .5, the array transform would be

1	0	0	0
0	.5	0	0
0	0	.5	0
0	0	0	.5

Multiplying an initial point (which had been changed to a row vector) of

	x	y	z
1	.300	.400	.867

by the array transform function would result in a new row-vector of

	x'	y'	z'
1	.150	.200	.433

The new x', y' and z' coordinates can then be tested to see whether or not they fit within the "Window in the Sky" (in this case they do not) and plotted, as required.

SUMMARY

As shown above, positioning and repositioning the stars in Starfinder ON DISPLAY is based upon

1. Assuming that the stars are equidistant from earth, at a distance of 1.
2. Changing the right-ascension and declination values to x, y, and z-coordinates.
3. Selecting an appropriate "Window in the Sky."
4. Plotting the stars, using their x and z-coordinates, if both are within the "Window" (and if the y-coordinate is positive).
5. Selecting characters to represent the stars, based upon their brightness (visual magnitude).
6. Using the array rotate transforms to move the stars up, down, left, right, clockwise, and counterclockwise.
7. Using the array scale transform to move the stars farther apart (zoom in) or closer together (back away).

With this approach, the user views the sky as they wish while moving the stars again and again. The screen is unhampered by clouds, the moon, streetlights, the time of day, the season of the year, or the physical location (on earth) of the viewer.

For a complete copy of the program write

Melvin O. Duke
P.O. Box 20836
San Jose, CA 95160
(408) 268-6637 □

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

CONFESSIONS OF A PASCAL PROGRAMMER

Reducing EXE File Size Through Data Positioning

by Daniel J. Jones

Often I hear people wondering why Pascal produces such large EXE files. I have discovered some "tricks" that reduce the size of the execution file. Presented here is one of them, which may help you reduce the size of some of your program files.

This particular trick involves placement of large data structures in the Pascal program. To begin, a quick review of the Pascal source program structure is in order. You will recall the structure is divided into three parts as follows:

(Program Header)
Program Name (Parameters)

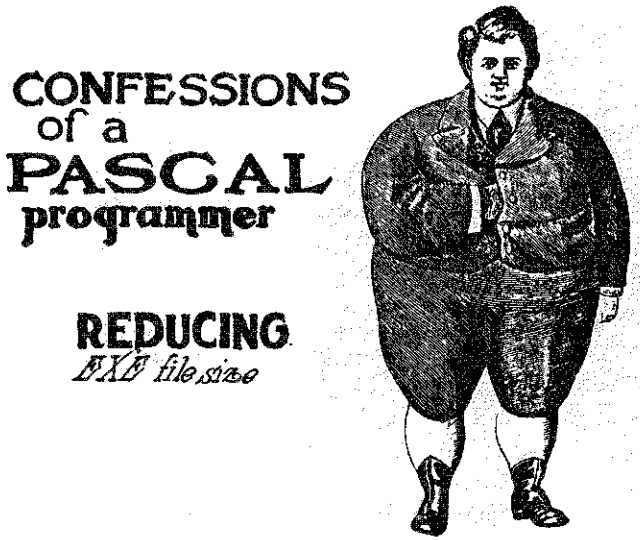
(Program Declarations)
Labels
Constants
Types
Variables
Procedures and Functions

(Program Body)
Begin
Pascal Statements
End.

I have composed a trivial example of a Pascal program. The task this program performs is unimportant. Its purpose is to illustrate the effects of large data structure placement on size and speed of the executable file. Here is the example Pascal program:

```
Program Example1;
Var     Large_structure :       Array [1..15000] of integer;
        i,j             :       integer;

Begin
For i := 1 to 100 do
    For j := 1 to 15000 do
        Large_structure[j] := 0;
End.
```

The variables in the program "Example1" above are defined at the program level. Program-level variables are automatically given the STATIC attribute. The STATIC attribute assigns a variable a fixed location in memory. Use of STATIC variables can save time and code space, but the trade-off is INCREASED DATA SPACE. This means, for many Pascal compilers, program-level variables have a direct effect on execution file size.



As shown in the Pascal structure above, the procedure is a program declaration. Since the procedure has a structure similar to the program, the example program above could be written like this:

```
Program Example2;

Procedure Main_procedure;
Var     Large_structure :       Array [1..15000] of integer;
        i,j             :       integer;
begin
For i := 1 to 100 do
    For j := 1 to 15000 do
        Large_structure[j] := 0;
end;

Begin
Main_procedure;
End.
```

The variables in the program example above are set up at the procedure level. Procedure level variables are either allocated on the stack or dynamically allocated on the heap. Therefore they are not given the STATIC attribute.

Simply by moving the data from the program-level to the procedure-level, the difference in the size of the EXE file is approximately 30,000 bytes. This is a significant reduction, but it can pose a problem.

Suppose the large data item is to be accessed by other loaded modules. This would require that the item have the PUBLIC attribute. Since the PUBLIC attribute implies a STATIC variable, how can the variable be allocated on the stack at procedure level and also be PUBLIC?

One answer is to use a reference type. Reference types include the pointer type and (in extended Pascal) the address type. Pointer types are usually portable while address types are often machine specific.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

If the address of the large data structure must be PUBLIC, the example program could be written using a pointer type like this:

```
Program Example3;

Var     [PUBLIC] Large_struc_ptr : ^Array [1..15000] of integer;
        i,j                      : integer;
Begin
New (Large_struc_ptr);
For i := 1 to 100 do
    For j := 1 to 15000 do
        Large_struc_ptr^[j] := 0;
Dispose (Large_struc_ptr);
End.
```

The PUBLIC variable in the program example above is a program-level item. It is a pointer variable which requires only four bytes and merely points to the actual data structure. The "NEW" procedure allocates the data item in the Pascal heap structure at run time. This keeps the large data variable from affecting the size of the executable file, while at the same time allowing the PUBLIC attribute.

Another method for allowing the PUBLIC attribute to point to a large data structure is to use the second pointer type, called the address type. If the Pascal extended features are available and portability is not a concern, the example program could be put together like this:

```
Program Example4;
Type    Large_struc_type          : array [1..15000] of integer;
Var     [PUBLIC] Large_struc_adr : adr of Large_struc_type;

Procedure Main_procedure;
        Large_struc               : Large_struc_type;
        i,j                       : integer;
begin
Large_struc_adr := adr Large_struc;
For i := 1 to 100 do
    For j := 1 to 15000 do
        Large_struc_adr^[j] := 0;
end;

Begin
Main_procedure;
End.
```

In the example above, the address type is a PUBLIC, program-level data variable of four bytes. The large data item is procedure level so is allocated on the stack. The address of the large data variable is assigned to the PUBLIC address variable at run time.

The example above shows use of the address variable to access the large data structure. Within the program "Example 4" the large data item can be accessed directly by the name "Large_struc", but access through the address type is shown as it would be done in an external procedure or function. The reason for this is to show an example using the address type, and also because the timing test will require the de-referencing of the PUBLIC address type.

The latter three example programs all show a method for keeping a large data variable from directly increasing the size of the executable program file. According to the Pascal documentation, this should increase code space and execution time. If so, by how much?

Using the debugger supplied with DOS, I decided to inspect the code that Pascal had compiled for each example. In each example program, the large data variable is filled with zeros one hundred times. With the exception of a little initialization, the only difference in the way these examples run is the method used to calculate the address to place the zero each time. Since this calculation method is the major difference in these programs, it is the main reason for the increase in code and speed of the program. Therefore, the statement to inspect is the assignment of zero to the large data structure. Here is the code I found for that assignment statement in each example:

```
Example1

    09AA:0019 8B3EF2F3      MOV     DI,[F3F2]
    09AA:001D D1E7          SHL     DI,1
    09AA:001F 81C7BE7E      ADD     DI,7EBE
    09AA:0023 C7050000      MOV     WORD PTR [DI],0000

Example2

    09AA:0014 8BBECC8A      MOV     DI,[BP+8ACC]
    09AA:0018 D1E7          SHL     DI,1
    09AA:001A 8D96CE8A      LEA     DX,[BP+8ACE]
    09AA:001E 03FA          ADD     DI,DX
    09AA:0020 C7050000      MOV     WORD PTR [DI],0000

Example3

    09AA:0025 8B3E04F4      MOV     DI,[F404]
    09AA:0029 D1E7          SHL     DI,1
    09AA:002B 033E00F4      ADD     DI,[F400]
    09AA:002F C745FE0000    MOV     WORD PTR [DI-02],0000

Example4

    09AA:001C 8BBECC8A      MOV     DI,[BP+8ACC]
    09AA:0020 D1E7          SHL     DI,1
    09AA:0022 033EF0F3      ADD     DI,[F3F0]
    09AA:0026 C745FE0000    MOV     WORD PTR [DI-02],0000
```

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Obviously the method of calculating the address to assign the zero to is different for each of the four examples. The size of these methods is summarized in the Table of Results below.

Next I ran all four of these programs on an IBM PC-type computer, timing the execution of each one. Presented here is a table of results to compare the examples:

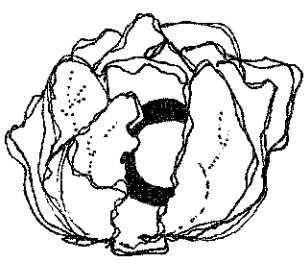
TABLE OF RESULTS

	Ex. 1	Ex. 2	Ex. 3	Ex. 4
EXE size (bytes)	49,908	19,940	19,938	19,940
Code size (bytes)	14	16	15	15
Time (sec)	51	59	57	61

In this table, the code size refers to the number of bytes required for the assignment statement, and the time indicates the number of seconds for the 100 repetitions of the zero fill process.

As you can see, nearly 30,000 bytes can be removed from the executable program "Example1" by using all three of the methods described above (Examples 2 through 4). That savings increases the coding of the data structure addressing computation by one to two bytes. Also, the time to execute three million repetitions of that addressing computation increases by six to ten seconds.

You should review your Pascal programs and discover if any large data variables are defined at program level. If so, and if your Pascal gives them the STATIC attribute, you should decide whether or not greatly reduced execution file size is worth a trade for slightly larger code size which implies slightly slower code. □



lettuce C

ADVANCED PROCESSORS ARE OK
TUNED SOFTWARE IS BETTER

Performance. Every user wants more of it.

Advanced hardware is one solution. But advanced hardware is difficult to obtain and expensive. Meanwhile, there are MILLIONS of existing machines waiting for a new generation of faster, more robust software. How will your software measure up?

Performance optimization should be considered part of the software development cycle.

Many programs spend 90% of the time executing less than 10% of the code. A small amount of time invested in optimizing these busy areas can yield big performance dividends. But rarely is it obvious just what pieces of source code account for 90% of the processor time. You need a sieve that can quickly isolate the big pieces (timewise) of your program.

CODE SIFTER is the tool you need to identify the time consuming sections of your EXE or COM file.

It observes your running program and generates a statistical report that indicates which areas of your program are prime candidates for your optimization efforts. The statistical output includes the symbols from your LINK map and can be directed to your display, printer, or to a file.

CODE SIFTER has advanced features.

Like a user friendly interface with function keys and online help. An adjustable sampling rate provides more accurate statistical data. An iteration option can rerun your program a number of times to reduce the busy areas to very narrow ranges. A demonstration program (with source) is provided for use with the step by step tutorial in the manual so that you can become productive in just a few minutes. CODE SIFTER works with any source language and does NOT require a knowledge of assembler to use. And if you are still not convinced – we will loan you a demonstration version.

David Smith Software
Box 25A R.D.#3
Oxford, N.Y. 13830
(607) 843–6209

$119

Requires
IBM–PC or XT
128K DOS 2.x

IBM is a registered trademark of International Business Machines Corporation

A FULL C COMPILER FOR $49.95

The Ecosoft Eco-C88 compiler for the 8088 and MSDOS is going to set a new standard for price and performance. Consider the evidence:

Compiler	Eco-C88	Lattice (1)	C86 (1)
Selve	13	11	13
Fib	44	58	46
Deref	13	13	--
Matrix	21	29	27
Price	$49.95	$500.00	$395.00

(1) **Computer Language,** Feb., 1985, pp.73-102. Reprinted by permission.

The Eco-C88 compiler is a full K&R C compiler that supports all data types and operators (except bit fields). Now look at the other features we offer:

* 8087 co-processor support using a single library. If you install an 8087 later, the software will use it without having to recompile.
* A robust standard library with over 150 functions, including transcendentals, color, and others.
* OBJ output for linking with the MSDOS linker (LINK).
* Error messages in English – no cryptic numbers to look up. A real plus especially if you're just getting started with C.
* Easy-to-read and complete user's manual.
* Works with all IBM and compatibles running MSDOS 2.0 (or later).
* Plus many other features.

For $10.00 more, we will include the source code for the C library functions (excluding transcendentals). For an additional $15.00, we will include our ISAM file handler in OBJ format (as published in the **C Programmer's Library,** Que Publishing). The discount prices for the library source and ISAM only apply at the time the compiler is purchased. Please add $4.00 to cover postage and handling. To order, call or write:

ECOSOFT INC.

Ecosoft Inc.
6413 N. College Avenue
Indianapolis, IN 46220
(317) 255-6476

MasterCard VISA AMERICAN EXPRESS

Eco-C (Ecosoft), MSDOS (Microsoft), UNIX (Bell Labs), CP/M (Digital Research), Z80 (Zilog), 8086, 8087, 8088 (Intel).

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TOOLS FOR BETTER PROGRAMS

by Terry W. Cox

Man is a tool-using animal, and the computer is a tool. It is, in fact, the ultimate tool — the tool that imitates thought. In the form of robots, the computer even becomes a tool that uses tools.

It stands to reason that those of us who program, who are reasonably familiar with computers and their uses, would make good use of the computer as a tool to assist us in writing software. Sadly, that's not always the case.

"Just a minute," says a voice in the audience. "Of course I use my PC when I program. I couldn't edit my programs or compile them without it, let alone run them."

And of course that's right. But the fact that we need a computer to edit, compile, and execute our programs doesn't mean that we're getting the maximum benefit from it. After spending hours trying to create a few lines of code to fill a particular need, or searching for an elusive bug, it's sometimes easy to think that computers are more of a hindrance than a help. We who use Pascal have an advantage in this regard since Pascal helps one write programs in a logical and systematic way. You might say Pascal is a better tool.

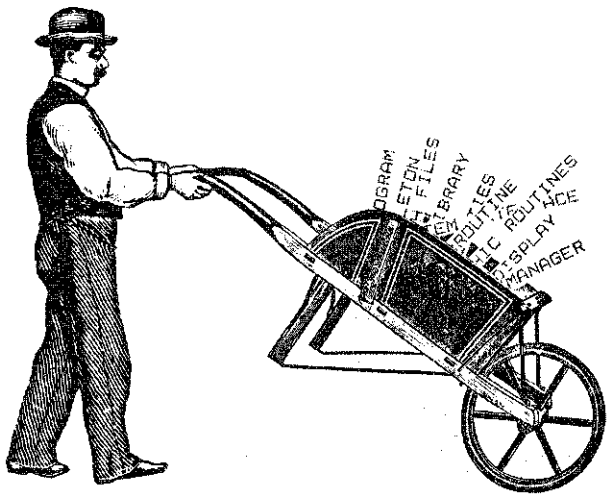
THE TOOLBOX

The right tool can speed up slow jobs and make complicated ones simple. Time spent building tools, as opposed to just coding, could be compared to money invested rather than money spent: it will continue to pay dividends for a long time. Let's look at some ways our PCs can be used to improve the quality and quantity of our programming.

You're probably using some of these tools, and some of the others will be familiar to you. Others may not be. The point to remember is that toolmaking is a behavior that, with practice, can become a habit. With the proper frame of mind you will constantly be on the lookout for tools and will begin finding them everywhere. Some will be free for the asking; others will cost you a few bucks (but will save you more than they cost, if they're the right ones).

SUBROUTINE LIBRARY

One of the most useful of programming tools, and also the most venerable, is a good subroutine library. Subroutines are the nuts and bolts of programming, as useful to the coder as small hardware is to the carpenter. A library is also a reliable barometer of its owner's attitude toward programming tools. There's the fellow who has a few subroutines lying about



here and there on one diskette or another, and who uses one when he happens across it; if software could rust, these routines would. Then there's the collector who grabs every routine he can get his hands on, but doesn't use (or in extreme cases even test) most of them. His library is as useful as a workshop full of new tools that are never used.

The best way to identify a good subroutine is the same way you'd tell which hand-tools are really useful: they're the worn ones with the sharp edges, the ones that get used. Here are some categories of subroutines that I've found generally useful.

SYSTEM INTERFACE

If there were just one subroutine available to a Pascal programmer, it should be a general system interface. This routine allows a program to access the facilities of DOS and BIOS calls. The routine SYSINT, at the bottom of this article, is an example of such a program. SYSINT is called with parameters that provide the interrupt code and the contents of the general registers. These values are passed to the interrupt. After the interrupt is executed, the register variables contain the contents of the registers as set by the DOS or BIOS call. By loading the register variables and interrupt-code variable with the appropriate values, Microsoft Pascal can perform any of the BIOS or DOS functions described in the DOS and Technical Reference manuals.

The two subroutines CLS and LOCATE in the program CURSOR (near the bottom of this article) are examples of routines built up from the system interface. CLS clears the screen; LOCATE positions the cursor at the specified row and column. These routines are also good examples of the other reason you should code or use subroutines: they hide the complexity

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

of the SYSINT call, with its required knowledge of DOS call parameters and register contents; in its place are simple, mnemonic, easy-to-use routines.

SCREEN DISPLAY/GRAPHIC ROUTINES

CLS and LOCATE provide convenient access to the full-screen capabilities of the PC display, but screen programming can be made much easier than this through the use of subroutines that allow the definition and manipulation of fields on the screen. Typical characteristics of each field are: position (row and column), length, attributes (e.g., blinking or reverse video), and whether it can be modified by the user. Commercial versions of screen managers usually provide a program to generate the field definitions interactively, but even without such embellishments a set of screen management subroutines will be among the most frequently used in your library. Graphic subroutines (for line-drawing, polygon-drawing and filling, and so on), while nice, tend to be less useful, since there are more occasions to display text than to use graphics.

Screen display and/or graphic routines can be thought of as what large system users call an "access method," which is just what its name suggests: a means of accessing a device — in this case the display. There are other access methods that are valuable but aren't provided "under the covers." Microsoft Pascal, for example, allows two ways to write to disk files: as text files, in which each record is terminated by a carriage return/linefeed, and as records of fixed length. The first approach allows records of variable length, while the fixed-length records allow retrieval of a particular record by record number. What isn't provided, which is assuredly worth having, is the ability to access records by key, an "indexed access method," or ISAM. Some of the more versatile commercial ISAMs include sophisticated buffer management to permit very fast record retrieval, multiple keys in the same file, and so on. If you aren't looking for advanced versions of these routines you can design your own or adapt versions supplied in various books. One such simple ISAM can be found in *The C Programmer's Library,* by Purdom, Leslie, and Stegemoller.

Beyond these generic families of subroutines, there are other useful subroutines that aren't so easy to classify, including the following:

* String-handling functions such as reversing a string or stripping leading or trailing blanks,

* Date routines that provide the current date in various formats; the day of the week; the number of days between two dates,

* A random-number generator; a routine to perform character set translation; various data type conversions such as character to hex or hex to integer, and

* Input validation routines for testing the data type of a string or range checking.

UTILITIES

A step up from the subroutine library are programs of general usefulness: "utilities." DOS comes with a large number of utilities, but not every useful program is on the system diskette. Programs that compress and expand disk files, that encrypt

TallScreen™ TAKES THE DRUDGERY OUT OF DOS

- 1000 lines of screen output saved to scroll through for review and editing
- Commands and directory paths automatically saved for quick recall again and again
- Screen output and commands edited with standard, powerful, easy-to-learn and easy-to-use word processor techniques
- Screen output saved to file or printer, or frozen anywhere on the screen
- DOS commands help file included for quick reference
- Keyboard macros supported similar to the IBM Personal Editor.

For the IBM PC®, PCjr, XT, AT, or compatibles. Works with DOS 2.00 and above. Works with PROKEY™ and SIDEKICK™. Compatible with most mouse systems. Next day shipping on VISA/MC. MD & NM residents add sales tax.

$49.95
INTRODUCTORY PRICE

"TallScreen has completely changed the way I use DOS."
Bob Beilstein, PC user

"A godsend for power users."
Hanno Hinsch, Author of TELIOS

"This is the way DOS should have been designed in the first place!"
Rex Swain, Technical Director Parallax Systems

"Just recalling screen output and past commands alone is worth the price."
Edward T. Garner, VP Finance ARIES Group

For orders or information see your local dealer or call 800-522-0290

QUALITAS, INC., P.O. BOX 3AK/UPB
LAS CRUCES, NM 88003 505-522-0290

QUALITY PROGRAMMING TOOLS
QUALITAS™

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

and decrypt data, that access files in special formats like word-processing files and spreadsheet data, are worth having. So are debugging aids that allow you to look at and modify your files, perhaps even patching a damaged disk file or directory (for example, the Norton Utilities). The book *Software Tools in Pascal,* by Kernighan and Plauger provides utilities such as filters, sort programs, and editing and formatting programs. This book is an excellent source of ideas on how to build programs from subroutines.

PROGRAM SKELETON

Another type of useful program is the program skeleton. A skeleton is the shell of a program that implements a particular algorithm, generalized and simplified so that it can be easily modified to work for any application that needs that algorithm. Every programmer borrows code from one program to use in another. Skeletons are simply an admission of this fact, and an attempt to make the best of it by insuring that the program you borrow code from is as good a model as it can be. Like utilities, the measure of how effective a skeleton is, is how often it's used. Thus, the skeletons you'll accumulate will depend largely upon the type of programming you do.

The combination of screen management and keyed record-access routines could eventually become part of a program that displays a record on the screen and allows you to update it. Since one record update looks much like another, such a program, when cleaned up, generalized, and condensed, would make a good skeleton. In a similar vein, a program that prints a formal report, with headings and such, makes a good skeleton, as does a file reformatting and copying program. Other candidates for skeletons are implementations of algorithms that are fairly complicated, such as a program that uses backtracking. Reference books on algorithms, like Donald Knuth's series of books *The Art of Computer Programming,* can provide many valuable skeletons.

BATCH FILES

Another category of aids, which are easy to make and use, are batch files. Practically everybody uses batch files, but a little care and effort can result in batch files that are "tuned" to your particular hardware configuration and, more important, to your working habits. We tend to issue the same sequences of commands over and over. Whenever you detect one of these sequences, create a batch file to take its place. I use batch commands for moving from one subdirectory to another and for compiling and linking modules.

If the batch command language isn't powerful enough for your needs, you might want to consider a replacement for the batch command processor called Extended Batch Language. It's marketed under the FREEWARE concept by Seaware Corporation, P.O. Box 1650, Delray Beach, FL 33444, and can be found on bulletin boards or ordered directly from Seaware for $30.

THE PROGRAM DEVELOPMENT ENVIRONMENT

My initial remarks, that every programmer uses the computer at least to compile his program, should be reexamined. Compilers have become increasingly sophisticated over the years, and that's especially true in the micro marketplace, with its overheated competition. This translates into opportunities for micro users. Compilers that generate more than object code are becoming common. One example, by now a classic, is the USCD Pascal system which includes a full-screen editor and provides some high-level debugging capabilities.

These features make USCD Pascal what has been called a "program development environment," in which the functions of editing a program and compiling to find errors are intermixed. Borland International's Turbo Pascal also includes a full-screen editor which is co-resident with the compiler and interacts with it. You use the editor to create or modify a program, which is kept in RAM. The compiler can then be invoked to compile the program, often while it's still in RAM, producing very rapid compiles. During compilation, if an error is detected, the compiler automatically invokes the editor, positioning the file to the offending source code statement.

A different type of tool is an "intelligent," or context-directed, editor. These are full-function editors that recognize the syntax of a particular language and give you warnings when you key in source statements that are incorrect. In some cases they won't even allow you to key in the wrong code.

Then there are programs that treat your programs as data, such as a cross-reference utility. Pascal can benefit from these programs more than other languages because of the simplicity and regularity of Pascal. This elegance allows the creation of programs that can automatically insert "white space" in a program to produce more readable listings. The process of using "white space" to improve readability is called prettyprinting. A good description of prettyprinting, as well as some practical reformatting rules, can be found in the book *Pascal With Style: Programming Proverbs* by Ledgard, Heuras, and Nagin. Another example of a program which treats programs as data is a case-converter. It is a form of prettyprinter that uses Pascal's ability to handle both upper- and lower-case identifiers. Typically, reserved words are converted to upper-case and user variables to lowercase, or vice versa, and, again, are designed to make a program easier to read. The sample program CURSOR was run through a prettyprinter and a case-converter as an illustration of what these programs can achieve.

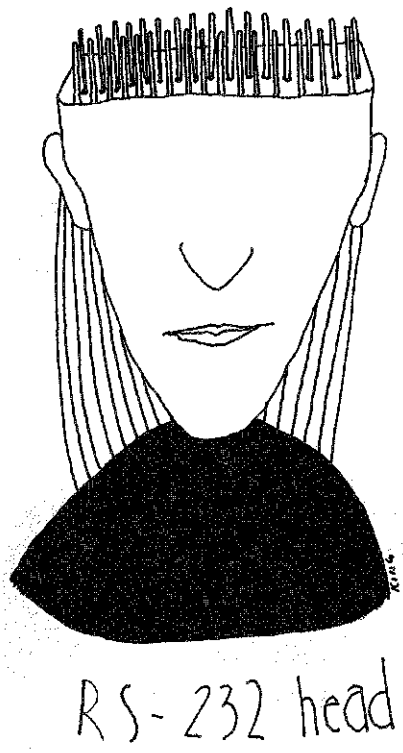
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

UPDATE MANAGER

The last category of software tools that I will mention are the programs that keep track of program development when multiple modules and/or multiple changes are involved. One such program is an "update manager," which controls multiple versions of a program, often by keeping just the updated text from one version to another. This can be a lifesaver when a bug is introduced into a program by a change you've made, since you can easily back out the change and examine just what the change affected as well. Other useful tools in this category are programs such as UNIX's MAKE, which determine those routines that are linked or included in various programs. This is useful when a subroutine is modified; then all programs using that subroutine can be identified and recompiled and/or linked.

CONCLUSION

The right tools can make a world of difference in programming, allowing us to program faster and with fewer errors. They can also reduce the frustrations of this complicated activity. There's an old motto that goes "A good workman never faults his tools." One of the reasons is that he usually has the right tools to start with.

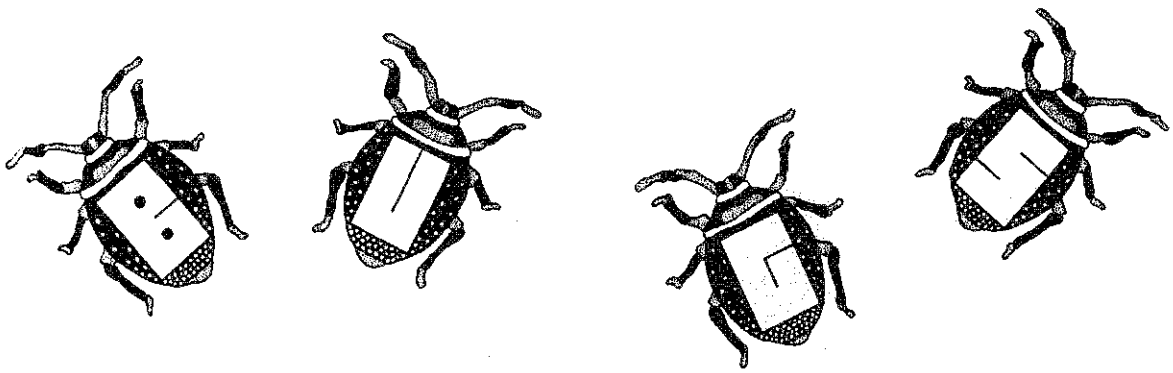



```
{$title: 'CURSOR CONTROL PROGRAM'}
{$LINESIZE: 132}
{$DEBUG+,$LINE+}

PROGRAM cursor (input, output);

{Here are a few bios codes we need for these routines }

CONST
      videoio     = 16#10;
      set_mode    = 16#00;
      bw80        = 16#02;
      cursor_pos  = 16#02;


VAR
    ax, bx, cx, dx: WORD;
    i: WORD;

{This declaration defines the external routine}

PROCEDURE sysint(intcode: WORD; VAR ax,bx,cx,dx: WORD); EXTERN;
{$page+}
{These routines are some examples of how sysint can be used}

PROCEDURE cls;
BEGIN
   ax := BYWORD(set_mode,bw80);
   sysint (videoio, ax, bx, cx, dx);
END;

PROCEDURE locate(row,column: INTEGER);
BEGIN
   ax := BYWORD(cursor_pos,0);
   dx := BYWORD(row,column);
   sysint (videoio, ax, bx, cx, dx);
END;


BEGIN
   cls;
   locate(1,10);
   write('Example subroutines CLS and LOCATE');
   locate(10,5);
   write('Can be useful');
   locate(10,30);
   write('In screen manipulation');
END.
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
page   54,130
; SYSINT  - Perform BIOS or DOS function
;
; Pascal calling sequence:
;
;     SYSINT(INTCODE: WORD; VAR AX,BX,CD,DX: WORD); EXTERNAL;
;
; AX-DX are four words which represent the general purpose registers.
; INTCODE is the desired system interrupt code.
;
; According to Pascal conventions, the stack looks like this on entry
; to the subroutine, with higher addresses at the top (note that the
; routine is entered via a far call).
;
;            SP + 0E:   top of frame (pointed to by BP)
;            SP + 0C:   INTCODE value
;            SP + 0A:   AX value
;            SP + 08:   BX value
;            SP + 06:   CX value
;            SP + 04:   DX value
;            SP + 00:   return address and segment (two words)
;
example segment byte public
        assume cs:example
        public sysint
;
sysint  proc   far
        push   bp                   ; save frame pointer
        mov    bp,sp                ; put our frame ptr in BP
        add    bp,02h               ; ..adjusted for the saved frame ptr
;
;       The following piece of code causes SYSINT to
;       modify itself.  The INT instruction at label
;       SYSINT2 is altered to the interrupt number
;       passed to the routine as INTCODE, so that when
;       the code at SYSINT2 executes, the correct
;       function is performed.
;
        call   sysint1              ; push the IP
sysint1:
        pop    ax                   ; put the IP in AX
        sub    ax,offset sysint1    ; calculate the relative address of
        add    ax,offset sysint2    ; the INT instruction (SYSINT2)
        mov    di,ax
        mov    ax,[bp+0ch]          ; get the desired interrupt number
        mov    cs:[di+1],al         ; put it in the INT instruction
;
;       Now load the registers and perform the interrupt.
;
        push   bp                   ; load the four registers from
        mov    bp,[bp+04h]          ; their variables.  since these
        mov    dx,[bp]              ; are variable parameters, their
        pop    bp                   ; addresses, rather than values,
        push   bp                   ; are passed on the stack, and they
        mov    bp,[bp+06h]          ; must be loaded indirectly.
        mov    cx,[bp]
        pop    bp
        push   bp
        mov    bp,[bp+08h]
        mov    bx,[bp]
        pop    bp
        push   bp
        mov    bp,[bp+0ah]
        mov    ax,[bp]
        pop    bp
        push   bp                   ; save BP across call
sysint2:
        int    00h                  ; perform the caller's request
        pop    bp                   ; restore BP value
;
;       Put the registers back into the passed parameter values
;       for the caller.
;
        push   bp                   ; store the four registers
        mov    bp,[bp+04h]          ; back into their respective
        mov    [bp],dx              ; variables.
        pop    bp
        push   bp
        mov    bp,[bp+06h]
        mov    [bp],cx
        pop    bp
        push   bp
        mov    bp,[bp+08h]
        mov    [bp],bx
        pop    bp
        push   bp
        mov    bp,[bp+0ah]
        mov    [bp],ax
        pop    bp
;
;       Clean up and exit.
;
        pop    bp                   ; get frame pointer
        ret    10                   ; get rid of the arguments and return
sysint  endp
;
example ends
        end  □
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



SYSTEMS IN ORGANIZATIONS, BUGS + FEATURES

review by J. Edward Volkstorf, Jr.

Data processing professionals are often called "problem solvers." The computer systems they design and implement are usually meant to provide a solution to a business inefficiency or make the business more competitive and so on. Unfortunately, as any seasoned dp veteran (and system user) knows, it does not always work out this way. Instead the system, that is "solution", often becomes a new problem for its users to contend with.

Why this happens so often is worth pondering, when you consider that systems are usually born from the desires of well-meaning users and business managers, aided by the good intentions of data processing experts. Furthermore, when you take into account the use of structured analysis, advanced programming tools, system design aids, program walk-throughs, user involvement schemes, and other such techniques, it would appear computer systems should not be as difficult to design, develop, and implement as they often turn out.

It is the contention of *Systems in Organizations, Bugs + Features* that a major problem which is often overlooked is the interaction of system features with the organizational structure in which the system is being used. Indeed, a strong case is developed that from the very inception of a new system its impact on the users from an organizational standpoint should be examined and planned for in the overall system's design.

The author, Lynne Markus, essentially states that the organizational environment can make or break the best of computer systems. Personnel relationships, job status, inter-organizational communications, work location, and other organizational issues will be replaced or at least changed with the new system to create potentially negative impacts if not properly planned for. Indeed, consider another viewpoint. Even a so called "lousy" system can function quite well in the proper organizational setting.

Ms. Markus is a consultant for Arthur D. Little, Inc., and obviously has drawn from a wide assortment of experiences in developing the conclusions found in her book. Extensive research is also evident by the large bibliography that is included. Numerous case studies and many short examples are described and analyzed using the issues developed by the book.

The current situation regarding traditional systems analysis techniques is also reviewed. Alternatives such as minimum critical specification and evolutionary development are discussed and contrasted, correlating them to the appropriate organizational structure.

A strong case is made against attempts at complete specification of a system before programming begins. Complete system specs are represented by a finished system in place and in use. The best approach is considered to be an iterative one with much back and forth communication all through the systems development process.

Relationships among users, vendors and the data processing "infrastructure" are each reviewed. This discussion provides some further insight into many of our current problems with computer systems.

Another interesting issue discussed is that of close user involvement. This is often considered a panacea to many dp problems, yet Ms. Markus shows that this is not always the case. She

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

describes the organizational climate and types of users which should and should not be closely involved in the system development process.

The book concludes with three recommendations regarding systems development:

1. Define the context and understand all relevant features in the organizational setting in which a system is being introduced. Carefully consider the goals and objectives of all users versus the goals and objectives of the system.

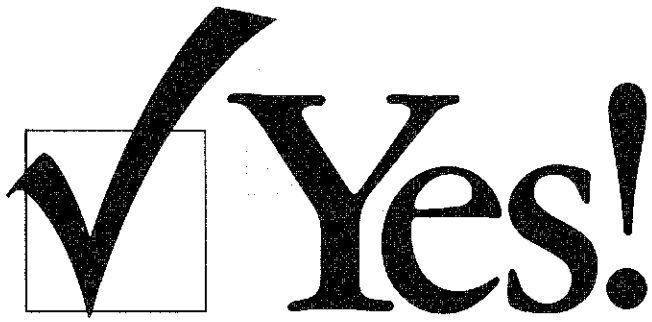
2. Eliminate or constrain system design features that are not politically, technically or culturally appropriate.

3. At the very beginning of a new system development or implementation specify the design process and the roles of all participants. Establish clear guidelines and define what is expected of all players, i.e., both the dp professionals and the users.

The above is best done by a strong leader in the organization at a high enough level to deal with all participants. Someone who can handle both the organizational problems of a system as well as the data processing related ones is needed. The book expresses a preference for someone in general management versus dp.

Systems in Organizations is very interesting reading. It fills a definite gap in the literature. If you have ever been involved in the design or implementation of a system then you will find something of interest. I found myself relating many of my personal experiences with the case studies presented. Most readers will be able to do the same. The book is quite rewarding for the insight and different perspective it provides. Anyone involved with computer systems in a medium to large organization will enjoy and learn something from reading this book.

Systems in Organizations, Bugs + Features by Lynne Markus Pitman Publishing Inc., Marshfield, MA, Copyright 1984. ☐



Please enter my new subscription to

programmer's journal

Bi-Monthly Subscription Rate

	U.S.	Canada (U.S. Funds only)	Foreign
1 year	☐ $24	☐ $30	☐ $40
2 years	☐ $44	☐ $55	☐ $75
3 years	☐ $64	☐ $80	☐ $110

Mail to: P.O. Box 3000 Department EE
Denville, NJ 07834

☐ New Subscription ☐ Check enclosed
☐ Renewal ☐ Bill me
☐ Charge my Bankcard: ☐ VISA ☐ MC

Card No. _____

Expiration date _____

_____
Signature

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

Telephone _____

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MISC15
NOBLINK CURSOR

If you are tired of the endlessly blinking cursor in various application packages then check out the products from Nostradamus. They have, among other offerings, a resident NoBlink utility that lets you establish the kind of cursor type you want: block, reverse video, background and foreground colors. There are 26 different combinations.

Nostradamus has a set of Turbo Pascal routines written in assembly that solve many interface problems. You can now fully utilize pathnames, file handles, hidden files, device independence, variable length records, etc.

NoBlink & Turbo Plus
Nostradamus
5320 So. 900 E. Suite 110
Salt Lake City, UT 84117
(801) 261-0769

MARKETING YOUR C PACKAGES

Advanced Product Solutions offers a complete range of retail consultation, marketing and distribution services to developers and manufacturers of computer products. APS is a retail distributor of software, specializing in products written in C that are transportable from IBM to the AT&T 3B2 and similar systems. They supply any marketing, merchandising, distribution and sales support for start-ups.

APS
100 Verona Court
Los Gatos, CA 95030
(408) 354-6583

EPSONS TO MACS

For $59.95 you can hook up your Mac to your trusty Epson FX-80 printer. The product is called "All You Need" and consists of a small printed circuit board that plugs into the serial port of your computer. If you are using a Mac, it responds to its codes, if you are using something else, it acts like a regular FX-80.

All You Need
Dresselhaus Computer Products
837 E. Alosta Ave.
Glendora, CA 91740
(818) 914-5831

HI-RES BORLAND

Fast graphics are now possible using Borland's Pascal in conjunction with their new Turbo Graphix Toolbox. It gives you high resolution graphics with window management. Real time animation is possible up to 500 frames a second as well as piecharts and business graphs. The toolbox is a set of procedures. You get the complete source code.

Turbo Graphix Toolbox
Borland International
4113 Scotts Valley Dr.
Scotts Valley, CA 95066
(408) 438-8400

C LANGUAGE GATHERING

C85 is the first international C Language conference. It will be held in San Francisco from May 13-15, 1985. Sponsors are The C Journal, Microsoft, Lifeboat Associates, and PC World. The conference is intended to be a non-commercial forum for software developers, programmers and others concerned with the future of C.

C85 will cater to all levels of C programmers. Tutorials and workshops will be held. Topics include portability, managing C projects, graphics, development tools, programming in MS-DOS/Mac/UNIX environments, debugging and compiler evaluation. The proposed draft of the ANSI X3J11 committee will be covered.

For further information:

C85 Planning Committee
c/o Lifeboat Assoc.
1651 Third Ave.
New York, NY 10128
(800) 457-4470

VDT MIX AND MATCH

If you find yourself wondering what it would be like to convert existing applications to run on a number of other terminals, then the book *A Programmers Guide to Video Display Terminals* may be a valuable tool, It lists the display attributes of over 150 terminals and gives you tips on how to handle the formidable task.

Programmers Guide to VDTs ($30)
Atlantis Publishing Corp.
P. O. Box 59467
Dallas, TX 75229
(214) 350-0986 □

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ARE YOU AN ISSUE SHY OF A FULL SET?

Complete your library! Order any back issues you may have missed — or become a new subscriber. You'll be glad you did.

- Vol. 1, No. 1: Norton Pascal I/O, Program Style Manual, COBOL, System Analysis, BASIC modules, Rounding error.
- Vol. 1, No. 2: Defining character sets, C86 BIOS calls, Copy protection, Pascal code, Input routines, Writing an Assembler.
- Vol. 1, No. 3: ASM dumb terminal, C source code, ASM graphics line drawing, Text editors reviewed, Utilities.
- Vol. 1, No. 4: Diskette directory via Pascal, MASM WHILE-WEND, C/ASM interface, ASM stay resident routine, Structured programming.
- Vol. 1, No. 5: Multi-tasking software review, Transporting C Programs, Programmers calculator, dBase bargraphs, Utilities.
- Vol. 1, No. 6: ASM printer configuration, DOS patch utility, Hi-speed graphics, Software ownership, dWINDOW review.
- Vol. 2, No. 1: Freelance consulting, Sorts in C, Pascal text editor source code, Designing national database, Ultra utilities.
- Vol. 2, No. 2: Understanding C through Assembly language, Utility routines in C and Assembly, Music in C (using a C music program), Cornucopia of software development tools.
- Vol. 2, No. 3: Software development environments, Creating a productive screen handling tool, Generating graphics within windows, Create, edit and play your musical ideas, Linking Assembler to Pascal and Basic.
- Vol. 2, No. 4: Writing a custom ROM BIOS for PC-DOS, The future of disk protection, Time-saving MASM Macros, C routines you can purchase.
- Vol. 2, No. 5: Scroll control using BASIC and Assembler, RTool, a PC-resident programmer's toolkit, Calling Assembly routines from dBase II, Pascal's pointer power.
- Vol. 2, No. 6: Periscope-Building a hardware board for the IBM PC, AT&T surprises all, dBase II to dBase III?, Intel 80286 bugs by FNE, calendar functions for BASIC.
- Vol. 3, No. 1: November's Comdex, C - its uses and abuses, Using the C preprocessor, Memory segmentation examples, Index files (part 1), Stuctured programming language, Periscope, a symbolic debugger.
- Vol. 3, No. 2: UNIX! (Again), Artificial Intelligence, Doctor C's Tips - Pointers to Functions, Calendar Date Routines in Basic, Calculate Record Layouts, Automatically!!

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

Telephone _____

Send me the following back issues: (Circle the issue(s) desired)
$5.50 each U.S., ($6.50 Canada, $8.50 Foreign, U.S. funds)
PRE-PAID ONLY

Vol. 1, No. 1	Vol. 1, No. 5	Vol. 2. No. 3	Vol. 3, No. 1
Vol. 1, No. 2	Vol. 1, No. 6	Vol. 2. No. 4	Vol. 3, No.2
Vol. 1, No. 3	Vol. 2, No. 1	Vol. 2. No. 5	
Vol. 1, No. 4	Vol. 2, No. 2	Vol. 2. No. 6	

Total amount of order _____

☐ Check enclosed

☐ Charge my Bankcard: ☐ VISA ☐ MC

_____ Expir. Date _____

Signature _____

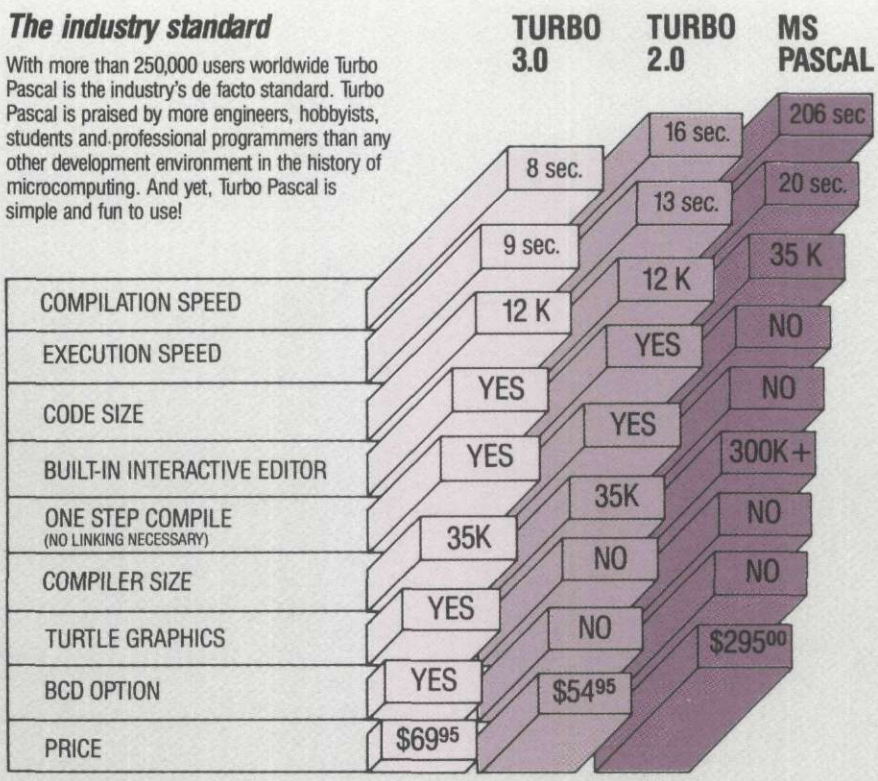
Send to:
PROGRAMMER'S JOURNAL
P.O. Box 30160
Eugene, OR 97403
(503) 484-2162

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

They said it couldn't be done. Borland Did It. Turbo Pascal 3.0

The industry standard

With more than 250,000 users worldwide Turbo Pascal is the industry's de facto standard. Turbo Pascal is praised by more engineers, hobbyists, students and professional programmers than any other development environment in the history of microcomputing. And yet, Turbo Pascal is simple and fun to use!



The best just got better: Introducing Turbo Pascal 3.0

We just added a whole range of exciting new features to Turbo Pascal:

- First, the world's fastest Pascal compiler just got faster. Turbo Pascal 3.0 compiles twice as fast as Turbo Pascal 2.0! No kidding.
- Then, we totally rewrote the file I/O system, and we also now support I/O redirection.
- For the IBM PC versions, we've even added "turtle graphics" and full tree directory support.
- For all 16 Bit versions, we now offer two additional options: 8087 math coprocessor support for intensive calculations and Binary Coded Decimals (BCD) for business applications.
- And much much more.

The Critics' Choice.

Jeff Duntemann, PC Magazine: *"Language deal of the century . . . Turbo Pascal: It introduces a new programming environment and runs like magic."*

Dave Garland, Popular Computing: *"Most Pascal compilers barely fit on a disk, but Turbo Pascal packs an editor, compiler, linker, and run-time library into just 29K bytes of random-access memory."*

Jerry Pournelle, BYTE: *"What I think the computer industry is headed for: well documented, standard, plenty of good features, and a reasonable price."*

Portability

Turbo Pascal is available today for most computers running PC DOS, MS DOS, CP/M 80 or CP/M 86. A XENIX version of Turbo Pascal will soon be announced, and before the end of the year, Turbo Pascal will be running on most 68000 based microcomputers.

An Offer You Can't Refuse

Until June 1st, 1985, you can get Turbo Pascal 3.0 for only $69.95. Turbo Pascal 3.0, equipped with either the BCD or 8087 options, is available for an additional $39.95 or Turbo Pascal 3.0 with both options for only $124.95. As a matter of fact, if you own a 16 Bit computer and are serious about programming, you might as well get both options right away and save almost $25.

Update policy

As always, our first commitment is to our customers. You built Borland and we will always honor your support.

So, to make your upgrade to the exciting new version of Turbo Pascal 3.0 easy, we will accept your original Turbo Pascal *disk* (in a bend-proof container) for a trade-in credit of $39.95 and your Turbo87 original *disk* for $59.95. This trade-in credit may only be applied toward the purchase of Turbo Pascal 3.0 and its additional BCD and 8087 options (trade-in offer is only valid directly through Borland and until June 1st, 1985).

(*) Benchmark run on an IBM PC using MS Pascal version 3.2 and the DOS linker version 2.6. The 179 line program used is the "Gauss-Seidel" program out of Alan R. Miller's book: *Pascal programs for scientists and engineers* (Sybex, page 128) with a 3 dimensional non-singular matrix and a relaxation coefficient of 1.0.

NOT COPY-PROTECTED

TURBOPASCAL

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by Credit Card call (800) 255-8008, CA (800) 742-1133

YES! I want the Best!

Please send: Quantity

Pascal 3.0 $ 69.95 ____

Pascal w/8087 $109.90 ____

Pascal w/BCD $109.90 ____

Pascal w/8087 & BCD $124.95 (SAVE $24.90) ____

*These prices include shipping to all U.S. cities. All foreign orders add $10 per product ordered.

Trade-in Credit Claimed: ____

Amount: (CA 6% tax) ____

Payment: VISA MC BankDraft Check

Credit Card Expir. Date: ____/____

Name on Card: ____

Card #: ____

Carefully Describe your Computer System!

Mine is: ___ 8 bit ___ 16 bit

I Use: ___ PC-DOS ___ MS-DOS ___ CP/M 80 ___ CP/M 86

My computers' name/model is: ____

The disk size I use is:

☐ 5¼" ☐ 8" ☐ 3½"

For update: original Turbo disk must accompany order

Name: ____

Shipping Address: ____

City: ____ Zip: ____

State: ____

Telephone: ____

COD's and Purchase Orders WILL NOT be accepted by Borland. California residents: add 6% sales tax. Outside USA: add $10 and make payment by bank draft, payable in US dollars drawn on a US bank.

36

BORLAND INTERNATIONAL

Software's Newest Direction

4113 Scotts Valley Drive
Scotts Valley, California 95066
TELEX: 172373

Turbo Pascal is a registered trademark of Borland International, Inc.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED